ITS Logical Architecture -Volume II
Process Specifications

Prepared by:

Lockheed Martin
Odetics Intelligent Transportation Systems Division

Prepared for:

Federal Highway Administration
US Department of Transportation
Washington, D. C. 20590

December 1999

TABLE OF CONTENTS

EDL# 5386 – “National ITS Architecture Documents: Communications Document; U.S. Department of Transportation”

EDL# 5388 - "National ITS Architecture Documents: Executive Summary; U.S. Department of Transportation" ;

EDL# 5389 - "National ITS Architecture Documents: Vision Statement: U.S. Department of Transportation" ;

EDL# 5390 - "National ITS Architecture Documents: Mission Definition; U.S. Department of Transportation" ;

EDL# 5391 - "National ITS Architecture Documents: Vol. 1. - Description; U.S. Department of Transportation" ;

EDL# 5392 - "National ITS Architecture Documents: Vol. 2. - Process Specifications; U.S. Department of Transportation" ;

EDL# 5393 - "National ITS Architecture Documents: Vol. 3. - Data Dictionary; U.S. Department of Transportation" ;

EDL# 5394 - "National ITS Architecture Documents: Physical Architecture; U.S. Department of Transportation" ;

EDL# 5395 - "National ITS Architecture Documents: Theory of Operations; U.S. Department of Transportation" ;

EDL# 5396 - "National ITS Architecture Documents: Traceability Matrix; U.S. Department of Transportation" ;

EDL# 5397 – “”National ITS Architecture Documents: Evaluatory Design; U.S. Department of Transportation”

EDL# 5398 – “National ITS Architecture Documents: Cost Analysis; U.S. Department of Transportation”

EDL# 5399 – “National ITS Architecture Documents: Performance and Benefits Study; U.S. Department of Transportation”

EDL# 5400 – “National ITS Architecture Documents: Risk Analysis; U.S. Department of Transportation”

EDL# 5401 – “”National ITS Architecture Documents: Evaluation Results; U.S. Department of Transportation”

EDL# 5402 – “National ITS Architecture Documents: Implementation Strategy; U.S. Department of Transportation”

EDL# 5403 - "National ITS Architecture Documents: Standards Requirements; U.S. Department of Transportation"

EDL# 5404 – “National ITS Architecture Documents: Standards Development Plan; U.S. Department of Transportation”

EDL# 11863 - "National ITS Architecture Documents: Market Packages - A Tool for Viewing, Accessing and Utilizing the National ITS Architecture; U.S. Department of Transportation".

ITS National Architecture

Document Control Information
CASE Model: 9/3/1999 @ 2:46:01 PM
Extracted: 9/3/1999 @ 2:46:01 PM
Logical Database: 09/03/99 @ 15:52:26

1.1.1.1 Process Traffic Sensor Data

Input Flows

fmmc_crossing_close_duration
fmmc_crossing_close_time
fp_pedestrian_data
fp_pedestrian_images
fre_physical_conditions
ft_traffic_data
ft_traffic_images
sensor_configuration_data

Output Flows

fault_data
hov_lane_data_input
hov_sensor_data
incident_analysis_data
local_sensor_data_for_highways
local_sensor_data_for_roads
multimodal_crossing_sensor_data
pedestrian_sensor_data
sensor_data_archive_input
sensor_data_for_reversible_lanes
sensor_fault_data
sensor_status
traffic_sensor_data
traffic_sensor_status
traffic_video_image
traffic_video_image_for_display

Description:

Overview: This process shall be responsible for collecting surveillance obtained from the roadside, vehicles, pedestrians (travelers using other modes of transport), railroad grade and multimodal crossings. Where any of the data is provided in analog form, the process shall be responsible for converting it into digital form and calibrating. The converted data shall be sent to other processes for distribution, further analysis and storage.

Data Flows: All inputs are received as solicited inputs as a result of its regular scan of data input sources and all outputs are solicited.

Functional Requirements: This process shall :
(a) continuously monitor the solicited data input flows shown above;
(b) where necessary convert the data obtained in (a) from analog to digital form, and calibrate the data;
(c) periodically send all of the surveillance data to other processes in the Manage Traffic function via the solicited output data flows shown above;
(d) complete a full scan of all inputs and generate the outputs in less than the time interval between successive activations.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.2;
USR = 1.6.2.1;
USR = 1.6.2.1.1;
USR = 1.6.2.2;
USR = 1.6.2.2.1;

USR = 1.6.2.3;
USR = 1.6.2.3.1;
USR = 1.6.2.4;
USR = 1.7;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(a);
USR = 1.8;
USR = 1.8.0;
USR = 1.8.3;
USR = 1.8.3.1;
USR = 1.8.3.1(b);

Output Flow Dynamics Assumptions:

hov_lane_data_input = 1;
incident_analysis_data = 1;
local_sensor_data_for_roads = 1;
local_sensor_data_for_highways = 1;
sensor_fault_data = 1;
sensor_data_for_reversible_lanes =1;
traffic_sensor_status = 1;
hov_sensor_data=1;
multimodal_crossing_sensor_data=1;
pedestrian_sensor_data=1;
traffic_sensor_data = 1;
traffic_video_image = 1;
traffic_video_image_for_display = 1;
sensor_data_archive_input = 1;
fault_data = 1;
sensor_status = 1;

1.1.1.2 Collect and Process Sensor Fault Data

Input Flows

environment_sensor_fault_data
environmental_sensor_data_store
environmental_sensor_status
fcm_sensor_fault_data
ftop_sensor_fault_data_input
sensor_fault_data
traffic_sensor_status

Output Flows

environmental_sensor_data_store
tcm_sensor_fault_data
ttop_current_sensor_faults

Description:

Overview: This process shall be responsible for collecting sensor status, identifying faults, and logging faults that have been detected by processes in other parts of the Manage Traffic function. It shall be possible for the faults to have been detected locally at the sensors, or centrally through communications links with the sensors. The process shall pass on new fault data to another processes for communication to the Construction and Maintenance terminator and shall receive fault clearances from the same terminator. It shall also maintain a store of the current fault state of all sensors. The process shall provide facilities that enable traffic operations personnel to review and update the current fault status of all sensors. Details of faulty and fixed equipment shall be passed by the process to the traffic control strategy selection process so that it can adjust its strategy to take account of the fault(s).

Data Flows: All input flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) the process shall be responsible for the maintenance of the store of the sensor fault data, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.4;
USR = 1.8.1.4(a);
USR = 1.8.1.5;
USR = 1.8.2;
USR = 1.8.2.13;
USR = 1.8.3;
USR = 1.8.3.1;

Output Flow Dynamics Assumptions:

ftop-sensor_fault_data_input = 1;
ttop-current_sensor_faults= 1;
fcm-sensor_fault_data = 1;
sensor_fault_data = 1;
tcm-sensor_fault_data = 1;

1.1.1.3 Process Environmental Sensor Data

Input Flows

environment_sensor_configuration_data
fre_environmental_conditions

Output Flows

archive_environmental_sensor_data
environment_sensor_data
environment_sensor_fault_data
environmental_sensor_status

Description:

Overview: This process shall be responsible for collecting data obtained from environmental sensors. Where any of the data is provided in analog form, the process shall be responsible for converting it into digital form and calibrating. The converted data shall be sent to other processes for distribution, further analysis and storage.

Data Flows: All inputs are unsolicited and all outputs are solicited.

Functional Requirements: This process shall :
(a) continuously monitor the solicited data input flows shown above;
(b) where necessary convert the data obtained in (a) from analog to digital form, and calibrate the data;
(c) periodically send all of the surveillance data to other processes in the Manage Traffic function via the solicited output data flows shown above;
(d) complete a full scan of all inputs and generate the outputs in less than the time interval between successive activations.

User Service Requirements:

USR = 1.0;
USR = 1.2;
USR = 1.2.3;
USR = 1.2.3.2;
USR = 1.2.3.2.3;
USR = 1.7;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(a);
USR = 1.7.1.1.1(b);
USR = 1.7.1.1.1(g);
USR = 1.7.1.2;
USR = 1.7.1.2.1;
USR = 1.7.1.2.1(b);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(e);

Output Flow Dynamics Assumptions:

environment_sensor_fault_data = 1;
environment_sensor_data = 1;
environmental_sensor_status = 1;
archive_environmental_sensor_data = 1;
.

1.1.1.4 Manage Data Collection and Monitoring

Input Flows

archive_environmental_sensor_data
fault_data
roadside_archive_control
roadside_data_archive
sensor_data_archive_input
sensor_status

Output Flows

roadside_archive_data
roadside_data_archive

Description:

Overview: This process shall collect and monitor sensor data from the roadside. The process shall collect the sensor data including sensor status and sensor faults from roadside equipment and distribute it to the Manage Archive Data function. The process shall run when a request for data is received from an external source.

Data Flows: All input data flows are unsolicited with the exception of roadside_archive_data and all output flows which are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited inputs shown above are received, the process shall immediately generate the solicited output shown above;
(c) data shall only be sent to the source from which the data request originated.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.1;
USR = 7.1.3.1.1(a);
USR = 7.1.3.1.1(c);
USR = 7.1.3.1.3;
USR = 7.1.3.1.3(e);
USR = 7.1.3.1.7;
USR = 7.1.3.1.7(a);
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(c);

Output Flow Dynamics Assumptions:

roadside_archive_data = roadside_archive_control;

1.1.2.1 Process Traffic Data for Storage

Input Flows

current_highway_network_data
current_incident_data
current_ramp_state
current_road_network_data
current_road_network_use
hov_lane_data
indicator_control_storage_data_for_highways
indicator_control_storage_data_for_roads
indicator_input_storage_data_for_highways
indicator_input_storage_data_for_roads
link_data_from_avl
link_data_from_tags
parking_lot_current_state
planned_event_data
processed_data
selected_strategy
sensor_output_data
vehicle_smart_probe_data_for_storage
wide_area_pollution_data

Output Flows

current_data
long_term_data

Description:

Overview: This process shall receive data from other processes and store the data into the long term and current data stores. The data shall comprise sensor data, both smoothed and unsmoothed: processed sensor surveillance data, data sent to control indicators (output devices e.g. intersection controllers, pedestrian controllers, dynamic message signs, ramp metering equipment), parking lot management data and other street equipment, the status data received from the indicators, plus current traffic conditions, planned events, current incidents, parking lot states, freeway ramp states, link travel times, roadway conditions provided by vehicle probes, and selected traffic control strategy. The data stored by the process in the current data store shall be the values collected over a relatively short period of time. The data stored in the long term data store shall be retained for a longer period. The data retained in the long term data store may be aggregated so as to reduce the storage requirements for long historical records, the amount of aggregation to be an implementation decision.

Data Flows: All input flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall :
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) maintain the store of current data in such a way that it contains data obtained over a limited time window, so that it presents a rolling picture of the current status and traffic conditions in the network, which is continually updated in real time;
(c) maintain the store of long term data in such a way that it contains the data from the current data store (optionally aggregated) to provide a complete historical record of the state of the system over a longer time window;
(d) the process shall be responsible for the maintenance of both current and long term data stores.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.2.5;
USR = 1.6.2.5.1;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.6;
USR = 1.8.1.6(f);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(b);
USR = 1.8.2.10;
USR = 1.8.2.10(c);

Output Flow Dynamics Assumptions:

current_data = 12/(60*60);
long_term_data = 4/(60*60);

1.1.2.2 Process Traffic Data

Input Flows

environment_sensor_data
hov_sensor_data
hri_sensor_data
multimodal_crossing_sensor_data
pedestrian_sensor_data
static_data_for_sensor_processing
traffic_sensor_data
traffic_video_image

Output Flows

parking_lot_input_data
processed_data
ramp_data
sensor_output_data
strategy_data_for_highways
strategy_data_for_roads
traffic_surveillance_data
unusual_data

Description:

Overview: This process shall receive and process data from sensors (both traffic and environmental) at the roadway. The process distributes data to Provide Device Control processes that are responsible for freeway, highway rail intersections, parking lot, surface street and freeway management. It also sends the data to another Provide Traffic Surveillance process for loading into the stores of current and long term data. Information about the various sensors to aid in this processing and distribution of data is accessed from the data store static_data_for_sensor_processing.

Data Flows: All inputs are unsolicited except for static_data_for_sensor_processing which is received as a result of requests for data retrieval. All outputs are solicited.

Functional Requirements: This process shall :
(a) run whenever any of the unsolicited input data flows listed above are received;
(b) use the data store 'static_data_for_sensor_processing' to analyze sensor data and determine how to allocate the received data to the various solicited output flows shown in (a) through (g) above, and send them to the appropriate processes in the Provide Device Control facility;
(c)analyze the input data to detect congestion and to pass this through the solicited output flow 'unusual_data' to the Manage Incidents facility;
(d) read data from the static data store 'static_data_for_sensor_processing'.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.2.2;
USR = 1.6.2.2.1;
USR = 1.6.2.3;
USR = 1.6.2.3.2;
USR = 1.6.2.4;
USR = 1.7.1.1;

Output Flow Dynamics Assumptions:

processed_data = 1;
parking_lot_input_data = 1;
ramp_data = 1;
strategy_data_for_highways = 1;
strategy_data_for_roads = 1;
unusual_data = 12/(60*60); USR = 1.6.2.4.1;
traffic_surveillance_data = 1; USR = 1.7;
sensor_output_data = 1; USR = 1.7.1;

1.1.2.3 Update Data Source Static Data

Input Flows

link_data_update
new_sensor_static_data
request_sensor_static_data

Output Flows

existing_sensor_static_data
link_details
static_data_for_sensor_processing

Description:

Overview: This process shall be responsible for the maintenance of the store of static data used in the processing of sensor data. This sensor data shall be used to provide traffic surveillance information for use by other processes within the Manage Traffic function. The store shall contain data showing the relationship between sensors and the surface street and freeway network, i.e. where they are located, to which part(s) of the network their data applies, the type of data, etc. It shall also hold information about the ownership of each link (that is, the agency or entity responsible for collecting and storing surveillance of the link) in the network which shall be used by processes involved in exchanging surveillance information (and) optionally control) with other Traffic Management Subsystems (TMS's). The contents of the store shall be provided by the Plan System Deployment function.

Data Flows: All inputs are unsolicited and all outputs are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) on receipt of 'link_data_update' or 'new_sensor_static_data'the process shall update the store of static data using the 'static_data_for sensor_processing' flow.
(c) on receipt of 'request_sensor_static_data', the process shall send the contents of the

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.2.4;
USR = 1.6.2.4.1;

Output Flow Dynamics Assumptions:

static_data_for_sensor_processing = 4/(60*60*24*7*52);
existing_sensor_static_data = 6/(60*60*24*7*52);
link_details = 1/(60*60*24);

1.1.2.4 Monitor HOV lane use

Input Flows

hov_lane_data_input
static_data_for_sensor_processing

Output Flows

hov_lane_data
hov_lane_violation

Description:

Overview: This process shall be responsible for monitoring the use of High Occupancy Vehicle (HOV) lanes and detecting vehicles that do not have the required number of occupants. The process also provides data on HOV lane usage for storage in the stores of current and long term data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'hov_lane_data_input'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval (the requests are implicit in the connection to a local database):
(a) 'static_data_for_sensor_processing'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'hov_lane_data';
(b) 'hov_lane_violation'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) process traffic sensor data to determine both the number of vehicles
using the HOV lane(s), the identity of vehicle(s) using the lane(s), and the passenger occupancy of vehicles using the lanes;
(c) the vehicle identities shall only be stored or issued in a data flow when the number of occupants in a vehicle is found to be less than that needed for the vehicle to legitimately use the HOV lane. This identity information shall be otherwise discarded without any residual storage;
(d) vehicle identities shall not be passed to the data storage process and shall only be sent to the Manage Emergency Services function in the event of an un validated and un verified violation being detected (law enforcement shall be responsible for validating and verifying these detected events);
(e) the process shall read data from the store of static data for the above processing.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.3;
USR = 1.6.3.4;
USR = 1.6.3.4(d);
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(b);
USR = 1.8.1.3;
USR = 1.8.1.3(b);
USR = 1.8.2;
USR = 1.8.2.4;
USR = 1.8.2.4(b);
USR = 1.8.2.11;
USR = 1.8.2.11(b);

Output Flow Dynamics Assumptions:

hov_lane_violation = 12/(60*60)*HOV_LANES;
hov_lane_data = 1;

1.1.2.5 Process Tag/AVL Data for Link Time Data

Input Flows

link_time_calculation_read
probe_data_for_traffic
static_data_for_sensor_processing
transit_probe_data
vehicle_tag_data

Output Flows

link_data_from_avl
link_data_from_tags
link_time_calculation_write

Description:

Overview: This process shall be responsible for processing tag and AVL data collected from roadside Readers or obtained from an analysis of toll transaction records. The process shall receive the processed tag and AVL data from the data collection process and shall calculate the travel time for the links under tag surveillance that have been traveled by the vehicles carrying the tags. This shall be achieved by noting the successive times at which the tag data is received and calculating the travel time from the difference. The data obtained from the toll tag transaction record analysis and/or tag reader locations shall not need any further processing as it shall contain the average travel times between successive toll collection plazas and tag reading locations. The process shall maintain a data store which contains the average travel time for each link in the surface street and freeway network under tag surveillance calculated from the previously described data. Calculation of the actual average values shall employ some type of aggregation processing (e.g. smoothing or similar technique) and be stored for differing time categories (e.g. times of day, day of week, holidays) in periodic increments. The current delay time for a link shall be the difference between current travel time value and the aggregate processed (e.g. average) value for that time category.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'probe_data_for_traffic';
(b) 'vehicle_tag_data';
(c) 'transit_probe_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of
requests for data retrieval:
(a) 'static_data_for_sensor_processing' - which contains data about links and tag reading points (the request is implicit in accessing data from this local data store);
(b) 'link_time_calculation_read' - which contains tag data (location ID, tag ID and a time stamp) received from reading tags at locations that have not been correlated with a subsequent tag measurement.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(a) 'link_data_from_tags';
(b) 'link_data_from_avl'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited data flows shown above;
(b) when the 'vehicle_tag_data' flow is received, store the tag data and check for any previous receipt from the same tag;
(c) if any previous occurrences of a particular tag data are found, compute the link travel time
(d) Assess if the value in (c) is realistic, i.e. it shall not be unduly long (an 'outlier') due to the vehicle being parked somewhere for a period of time. For example, this assessment could be made by comparing the travel time with the travel time value for the time of day already held in the data store or with other recently received data for the same link (also known as 'outlier analysis');
(e) if the new value is not assessed to be an outlier, then use it to update the aggregate (e.g. average) link travel time for the current time category and update the data store;
(f) if the new value shows a significant difference from the stored value (and is not assessed to be an outlier) then update an estimate of the link delay time;
(g) following completion of (e) or (f), generate the output of link travel and delay times, setting the values for any links where there is no travel time data to zero (0) to indicate 'no or uncertain data';
(h) the process shall be responsible for the maintenance of the store of link calculation data;
(i) the process shall read data from the store of static data in support of its analysis.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.2;
USR = 1.6.2.2;
USR = 1.6.2.4;
USR = 1.6.2.4.1;

USR = 1.6.2.5.1;

Output Flow Dynamics Assumptions:

link_data_from_tags = 12/(60*60);
link_time_calculation_write = 12/(60*60);
link_data_from_avl = 12/(60*60);

1.1.2.6 Process Collected Vehicle Smart Probe Data

Input Flows

vehicle_smart_probe_input_data

Output Flows

vehicle_smart_probe_data_for_storage
vehicle_smart_probe_data_indication

Description:

Overview: This process shall be responsible for the processing of vehicle smart probe data. The process receives data from vehicle subsystems and processes the data to estimate type and level of roadway conditions and hazards. The process shall send the road condition and hazard estimates to the Provide Device Control facility for output to future passing vehicles. It shall send this data, together with the fixed unit identity and fixed location to the traffic data storage process for loading into the current and long term data stores.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'vehicle_smart_probe_input_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(a) 'vehicle_smart_probe_data_for_storage';
(b) 'vehicle_smart_probe_data_indication'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited data flow 'vehicle_smart_probe_input_data';
(b) when the data flow in (a) is received, use it to estimate road conditions and hazards;
(c) send the processed road condition and hazard estimate data to the vehicle smart probe data output process in the Provide Device Control facility;
(d) combine the road condition and hazard estimate data with the fixed unit identity and location and send it to the traffic data storage process; (Fixed unit identity and location shall be initialized when the process is initiated.)

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.2;
USR = 1.6.2.2;
USR = 1.6.2.3;
USR = 1.6.2.3.1;
USR = 1.6.2.3.2;
USR = 1.6.2.4;
USR = 1.6.2.5;
USR = 1.8.2;
USR = 1.8.2.10;
USR = 1.8.2.10(b);
USR = 1.9;
USR = 1.9.0;
USR = 1.9.2;
USR = 1.9.2.1;
USR = 1.9.2.1.3;

Output Flow Dynamics Assumptions:

vehicle_smart_probe_data_indication = vehicle_smart_probe_input_data;
vehicle_smart_probe_data_for_storage = vehicle_smart_probe_input_data;

1.1.2.7 Monitor Reversible Lanes

Input Flows

reversible_lane_status
reversible_lane_video_images
sensor_data_for_reversible_lanes
static_data_for_sensor_processing

Output Flows

wrong_way_vehicle_detection

Description:

Overview: This process shall be responsible for monitoring the use of reversible lanes and detecting wrong-way
vehicles in reversible lanes. The process shall monitor sensor data and video images from the reversible lanes, and use this information along with the lane status (which direction it is currently operating) to identify when a vehicle is traveling in the wrong direction on the reversible lane.

(a) 'static_data_for_sensor_processing';
(b) 'incident_video_images'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval (the requests are implicit in the connection to a local database):
(a) 'sensor_data_for_reversible_lanes'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'wrong_way_vehicle_detection'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) process traffic sensor data to determine both the number of vehicles
detected in reversible lane(s) and the identity of vehicle(s) using the lane(s);
(c) vehicle identities shall not be passed to the data storage process and shall be sent to the traffic operations personnel and the Manage Emergency Services function in the event of an unvalidated and unverified violation being detected (law enforcement shall be responsible for validating and verifying these detected events).

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(a);
USR = 1.7.4;

Output Flow Dynamics Assumptions:

wrong_way_vehicle_detection = 12/(60*60);

1.1.3 Generate Predictive Traffic Model

Input Flows

current_incident_data
fws_predicted_weather
long_term_data
other_traffic_center_data
planned_events
predictive_model_data
selected_strategy

Output Flows

prediction_data
predictive_model_data
unusual_congestion

Description:

Overview: This process shall be responsible for continually producing and updating a predictive model of the traffic flow conditions in the road or freeway network served by the Manage Traffic function that an instance of this process is allocated to. The prediction shall be based on current surveillance, historic traffic data and surveillance, current incidents, planned events, current traffic control strategy, data received from other Traffic Management Subsystems (TMS's) serving other geographic and/or jurisdictional areas, and current and predicted weather conditions. The predictive model of traffic flow produced by this process shall be used by processes in the Manage Traffic function and other ITS functions.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'current_incident_data';
(b) 'fws-predicted_weather';
(c) 'planned_events';
(d) 'selected_strategy'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'historical_data';
(b) 'other_traffic_center_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'prediction_data';
(b) 'predictive_model_data';
(c) 'unusual_congestion'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above storing the received data internally to the process;
(b) periodically or continuously produce an updated predictive estimate of the traffic flow conditions within the road network served by the specific instance of the Manage Traffic function, identifying any segments on which unusual congestion will form;
(c) the process shall be responsible for the maintenance of the store of predictive data.

User Service Requirements:

USR = 1.0;
USR = 1.2;
USR = 1.2.3;
USR = 1.2.3.2
USR = 1.2.3.2.3;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.2.5;
USR = 1.6.2.5.2;

Output Flow Dynamics Assumptions:

predictive_model_data = 4/(60*60);
prediction_data = 4/(60*60);
unusual_congestion = 4/(60*60);

1.1.4.1 Retrieve Traffic Data

Input Flows

current_data
long_term_data
predictive_model_data
request_traffic_media_data
request_traffic_operations_data
traffic_data_demand_request
traffic_data_deployment_request
traffic_data_distribution_request

Output Flows

incident_video_for_emergency_services
operator_log_for_traffic_data
retrieved_traffic_media_data
retrieved_traffic_operations_data
sensor_data_for_distribution
traffic_data_for_demand
traffic_data_for_deployment
traffic_data_for_distribution
traffic_data_for_emergency_services
traffic_data_for_signage
traffic_data_for_transit

Description:

Overview: This process shall on request retrieve traffic data from the data stores managed by other processes in the Provide Traffic Surveillance facility of the Manage Traffic function. It shall be possible for requests to originate from traffic operations personnel, the media operator, the Manage Demand facility within the Manage Traffic function, the Plan System Deployment function and the Provide Driver and Traveler Services function. With the exception of those from the Manage Demand facility and the Plan System Deployment function, all requests shall be provided by interface processes. The process shall also generate traffic data for output by other processes to in-vehicle signage functions.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_traffic_operations_data';
(b) 'traffic_data_distribution_request';
(c) 'traffic_data_demand_request';
(d) 'request_traffic_media_data';
(e) 'traffic_data_for_deployment'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'current_data';
(b) 'long_term_data';
(c) 'predictive_model_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'retrieved_traffic_operations_data';
(b) 'traffic_data_for_demand';
(c) 'traffic_data_for_distribution';
(d) 'traffic_data_for_signage';
(e) 'traffic_data_for_transit';
(f) 'retrieved_traffic_media_data;'
(g) 'incident_video_for_emergency_services;'
(h) 'operator_log_for_traffic_data';
(i) 'traffic_data_deployment_request'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited data flows listed above;
(b) when any of the flows in (a) are received, retrieve data from the current, long term and predictive model data stores, using the solicited input data flows listed above;
(c) generate and issue the solicited output flow listed above that corresponds to the input flow, loading into it the data appropriate to the recipient;
(d) periodically or on an event driven basis generate the data flow sent to the traffic control

process responsible for sending data to processes that broadcast to in-vehicle signage equipment;
(e) the process shall retrieve data from the stores of current, long term and predictive data as needed to support its other processing requirements.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.3;
USR = 1.6.3.4;
USR = 1.6.3.4.1;
USR = 1.6.4;
USR = 1.6.4(a);
USR = 1.6.4(b);
USR = 1.6.4(c);
USR = 1.6.4(d);

Output Flow Dynamics Assumptions:

retrieved_traffic_operations_data = request_traffic_operations_data;
retrieved_traffic_media_data = request_traffic_media_data;
sensor_data_for_distribution = 1/(60);
traffic_data_for_demand = traffic_data_demand_request;
traffic_data_for_distribution = traffic_data_distribution_request;
traffic_data_for_signage = 12/(60*60);
traffic_data_for_transit = 12/(60*60);
incident_video_for_emergency_services = request_traffic_operations_data;
traffic_data_for_emergency_services = request_traffic_operations_data;
traffic_data_for_deployment = traffic_data_deployment_request;

1.1.4.2 Provide Traffic Operations Personnel Traffic Data Interface

Input Flows

ftop_traffic_data_parameter_updates
ftop_traffic_information_requests
ftop_weather_request_information
map_data_for_traffic_display
operator_log_for_traffic_data
retrieved_traffic_operations_data
traffic_video_image_for_display
weather_service_information

Output Flows

environment_sensor_configuration_data
operator_log_for_traffic_data
request_traffic_map_display_update
request_traffic_operations_data
sensor_configuration_data
traffic_data_media_parameters
ttop_traffic_control_information_display
ttop_video_image_output
ttop_weather_information
weather_service_information_request

Description:

Overview: This process shall provide the interface through which traffic operations personnel can obtain access to the data stored by other processes in the Provide Traffic Surveillance facility of the Manage Traffic function, and set up the parameters that govern the data that is available to non-traffic operations people via a separate process to the media operator. This stored data shall comprise current and long term (historic) data on traffic conditions, weather conditions and roadside equipment activity, plus prediction estimates of traffic conditions. The data shall apply to some or all of the surface street and freeway network served by the specific instance of the Manage Traffic function. Where appropriate and/or requested by the traffic operations personnel, the process shall provide the data output in the form of an overlay onto a map of the relevant part(s) of the surface street and freeway network served by the instance of the function. The process shall obtain the map from

a local data store, which it shall enable the traffic operations personnel to update as and when required.

all outputs, is solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor the input data flows and provide acknowledgment of receipt through a human interface of those from traffic operations personnel;
(b) be capable of carrying out its own verification of input data received from traffic operations personnel and generating the correct solicited output data flow as a result of input data being received;
(c) as part of the output generation process, carrying out checks for data out of range, missing or containing spurious values and requesting re-input where required;
(d) be capable of simultaneously handling multiple independent input/output data channels, i.e. supporting access by more than one traffic operations personnel;
(e) providing all output to traffic operations personnel in a form that is readily understood by a human operator;
(f) only generate the outputs listed above as a result of receiving inputs from the traffic operations personnel or other processes;

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.1;
USR = 1.6.1.7;
USR = 1.6.1.7(a);
USR = 1.6.3;
USR = 1.6.3.4(e);

Output Flow Dynamics Assumptions:

request_traffic_operations_data = ftop-traffic_information_requests;
request_traffic_map_display_update = 4/(60*60*24*7*52);
ttop-traffic_control_information_display = ftop-traffic_information_requests;

```
ttop-video_image_output = ftop-traffic_information_requests;
traffic_data_media_parameters = 1/(60*60*24*7);
sensor_configuration_data= 1/(60*60*24);
weather_service_information_request = weather_service_traffic_request;
ttop-weather_information = ftop-weather_request_information;
environment_sensor_configuration_data = 1/(60*60*24*7);
```

1.1.4.3 Provide Direct Media Traffic Data Interface

Input Flows

fm_traffic_data_request
map_data_for_traffic_display
retrieved_traffic_media_data
traffic_data_media_parameters

Output Flows

request_traffic_media_data
tm_traffic_data

Description:

Overview: This process shall be responsible for providing the interface between the media and the process responsible for obtaining data from the stores of traffic data maintained by other processes within the Provide Traffic Surveillance facility of the Manage Traffic function. The process shall enable the media to request and be provided with current, long term (historic) and predicted traffic data. The data may be provided in one or more formats: as a data stream, as processed and displayed to Traffic Operations Personnel (e.g. graphical summaries of link speeds),or as a display (with data included on a map of relevant part(s) of the road and freeway served by the Manage Traffic function. The media shall only be able to request and see displayed that data that the traffic operations personnel have made available, through the use of the definition in the traffic data media parameters.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fm-traffic_data_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of
output being sent to other processes and requests for data retrieval from local data stores:
(a) 'map_data_for_traffic_display';
(b) 'retrieved_traffic_media_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of
the above inputs being received:
(a) 'request_traffic_media_data';
(b) 'tm-traffic_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor the input data flows and provide acknowledgment of receipt of those from the media;
(b) be capable of accepting input from the media in audio or other forms, where the latter
may comprise input from any combination of keyboards or other forms of push-button devices, pointing
devices, etc.;
(c) be capable of carrying out its own verification of input data received from the media
and generating the correct solicited output data flow as a result of input being received;
(d) as part of the output generation process, carrying out checks for data out of range, missing or
spurious values and requesting re-input where necessary;
(e) be capable of simultaneously handling a large number of independent input/output data channels,
i.e. supporting very many media, some of whom may be remote;
(f) providing all output to the media in a form that is readily understood by a human
operator and which may be in audio or visual form, with the latter being available in a variety of
formats, e.g. displays, or hardcopy (paper) output;
(g) only generate the outputs listed above as a result of receiving inputs from the media
or the other processes;
(h) the use of the digitized map display shall be automatic and shall be at a resolution best suited
to the quantity and scope of data being displayed, i.e. the map shall be to the largest possible scale.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.1;
USR = 1.6.1.7;
USR = 1.6.1.7(a);
USR = 1.7;
USR = 1.7.0;
USR = 1.7.3;
USR = 1.7.3.3;

Output Flow Dynamics Assumptions:

request_traffic_media_data = (1/60)*MEDIA_OPS;
tm-traffic_data = (1/60)*MEDIA_OPS;

1.1.4.4 Update Traffic Display Map Data

Input Flows

fmup_traffic_display_update
request_traffic_map_display_update

Output Flows

map_data_for_traffic_display
tmup_request_traffic_display_update

Description:

Overview: This process shall provide updates to a store of digitized map data when a request is received from traffic operations personnel via their interface process. The map data shall be for use as the background for displays of traffic data requested by traffic operations personnel and media operators through their respective interface processes. This process shall obtain the new map data from either a specialized data supplier or some other appropriate data source.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'request_traffic_map_display_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to external functions:
(a) 'fmup-traffic_display_update'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmup-request_traffic_display_update';
(b) 'map_data_for_traffic_display'.

Functional Requirements: This process shall:
(a) continuously monitor for the receipt of the unsolicited data flow shown above;
(b) when the data flow 'request_traffic_map_display_update' is received, generate the 'tmup-request_traffic_display_update' output data flow and continuously monitor for receipt of the solicited input data flow 'fmup-traffic_display_update';
(c) when the flow 'fmup-traffic_display_update' is received, prepare and output the 'map_data_for_traffic_display' data flow;

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.3;
USR = 1.6.3.4(e);
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
URS = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(e);

Output Flow Dynamics Assumptions:

map_data_for_traffic_display = request_traffic_map_display_update;
tmup-request_traffic_display_update = request_traffic_map_display_update;

1.1.4.5 Provide Media System Traffic Data Interface

Input Flows

fm_incident_details
fm_incident_information_request
fm_traffic_information_request
information_for_media

Output Flows

incident_details_from_media
tm_incident_information
tm_traffic_information

Description:

Overview: This process shall provide the interface through which traffic and incident data can be output to the Media. The output shall comprise traffic and incident data that is suitable for output to the Media System as determined by traffic managers. This interface is only for the output of data that has been requested by the Media.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the 'information_for_media' data flow;
(b) when received convert information_for_media into a form for output to the media.

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

tm-traffic_information = information_for_media;
incident_details_from_media = fm-incident_details;
tm-incident_information = information_for_media;

1.1.4.6 Provide Traffic Data Retrieval Interface

Input Flows

current_traffic_pollution_data
incident_details_from_media
sensor_data_for_distribution
traffic_data_advisory_request
traffic_data_for_distribution
traffic_data_guidance_request
traffic_data_kiosk_request
traffic_data_personal_request
traffic_data_retrieval_parameters
traffic_data_ridesharing_request
traveler_traffic_profile

Output Flows

information_for_media
traffic_data_distribution_request
traffic_data_for_advisory_output
traffic_data_for_broadcast_to_kiosks
traffic_data_for_broadcast_to_personal_devices
traffic_data_for_guidance
traffic_data_for_kiosks
traffic_data_for_personal_devices
traffic_data_for_ridesharing
traffic_data_kiosk_request_for_archive
traffic_data_personal_request_for_archive
traffic_data_retrieval_parameters

Description:

Overview: This process shall provide customized sets of traffic data for broadcast, advisories, and personalized data to travelers, traveler information data archive, and the media. This process shall use the parameters in the data store 'traffic_data_retreival_parameters' to define exactly what data shall be retrieved as a result of each request. The process shall select the appropriate subset of traffic data which will be sent to each ITS function which is requesting data. The process shall accept traveler profiles for use in determining what personalized data to send to the traveler. The process shall send kiosk and personal traffic requests to the archival process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'incident_details_from_media';
(b) 'traffic_data_advisory_request';
(c) 'traffic_data_guidance_request';
(d) 'traffic_data_personal_request';
(e) 'traffic_data_kiosk_request';
(f) 'traffic_data_portables_request';
(g) 'traffic_data_ridesharing_request';
(h) 'traveler_traffic_profile';
(i) 'current_traffic_pollution_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'traffic_data_retrieval parameters';
(b) 'traffic_data_for_distribution';
(c) 'sensor_data_for_distribution'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'information_for_media';
(b) 'traffic_data_distribution_request';
(c) 'traffic_data_for_advisory_output';
(d) 'traffic_data_for_guidance'
(e) 'traffic_data_for_kiosks';
(f) 'traffic_data_for_portables';
(g) 'traffic_data_for_ridesharing';

(h) 'traffic_data_personal_request_for_archive';
(i) 'traffic_data_kiosk_request_for_archive';
(j) 'traffic_data_for_broadcast_to_kiosks';
(k) 'traffic_data_for_broadcast_to_personal_devices'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of any of the unsolicited input data flows listed above;
(b) when the flow received in (a) is a request for data, send the request to the data retrieval process and data archival process using the request solicited output data flow shown above;
(c) when the response to the request flow in (b) is received, assemble the data for output according to the data in the 'traffic_data_retrieval_parameters' data store;
(d) when (c) is complete, send the retrieved data to the requesting process in the corresponding solicited output flow shown above;
(e) if the data flow in (a) contains new data for the store of traffic data retrieval parameters, load it into the 'traffic_data_retreival_parameters' data store.

User Service Requirements:

USR = 1.0;
USR = 1.1;
USR = 1.1.4;
USR = 1.1.4.1;
USR = 1.1.4.1.1;
USR = 1.1.4.1.2;
USR = 1.1.4.1.3;
USR = 1.1.4.1.4;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.3;
USR = 1.6.3.4;
USR = 1.6.3.4.1;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(g);

Output Flow Dynamics Assumptions:

traffic_data_retrieval_request = traffic_data_advisory_request+traffic_data_kiosk_request+traffic_data_portables_request+ traffic_data_guidance_request+ traffic_data_ridesharing_request;
traffic_data_for_advisory_output = traffic_data_advisory_request;
traffic_data_for_guidance = traffic_data_guidance_request;
traffic_data_for_kiosks = traffic_data_kiosk_request;
traffic_data_for_personal_devices = traffic_data_personal_request;
traffic_data_for_ridesharing = traffic_data_ridesharing_request;
traffic_data_retrieval_parameters = traffic_data_media_parameters;
information_for_media = traffic_data_for_retrieval;
traffic_data_distribution_request = traffic_data_for_distribution + sensor_data_for_distribution;
traffic_data_personal_request_for_archive = traffic_data_personal_request;
traffic_data_kiosk_request_for_archive = traffic_data_kiosk_request;
traffic_data_for_broadcast_to_kiosks = traffic_data_kiosk_request;
traffic_data_for_broadcast_to_personal_devices = traffic_data_personal_request;

1.1.4.7 Manage Traffic Archive Data

Input Flows

ahs_operational_data
ftop_archive_command
static_data_for_archive
traffic_data_archive
traffic_data_for_deployment
traffic_management_archive_request
traffic_management_archive_status

Output Flows

traffic_data_archive
traffic_data_deployment_request
traffic_management_archive_data
ttop_archive_status

Description:

Overview: This process shall collect traffic data and ahs operational data to distribute to the Manage Archive Data function. The process shall run when a request for data is received from an external source, or when fresh data is received.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'archive_traffic_data_for_deployment';
(b) 'traffic_management_archive_request';
(c) 'ahs_operational_data';
(d) 'traffic_data_for_deployment';
(e) 'static_data_for_archive';
(f) 'ftop-archive_command';
(g) 'traffic_management_archive_status'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'traffic_data_archive';
(b) 'traffic_management_archive_status'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'traffic_management_archive_data';
(b) 'traffic_data_deployment_request';
(c) 'ttop-archive_status'.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.1;
USR = 7.1.3.1.1(a);
USR = 7.1.3.1.1(b);
USR = 7.1.3.1.1(c);
USR = 7.1.3.1.1(d);
USR = 7.1.3.1.1(e);
USR = 7.1.3.1.2;
USR = 7.1.3.1.3;
USR = 7.1.3.1.3(e);
USR = 7.1.3.1.5;
USR = 7.1.3.1.5(e);
USR = 7.1.3.1.5(g);
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(b);
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(a);

Output Flow Dynamics Assumptions:

traffic_management_archive_data = traffic_management_archive_data_request;

traffic_data_deployment_request = ftop-archive_command;
ttop-archive_status = ftop-archive_command;

1.1.5 Exchange data with Other Traffic Centers

Input Flows

control_data_for_highways
control_data_for_roads
current_data
cv_incidents_for_other_TMC
emergency_data_for_other_TMC
fotc_data_request
fotc_identity
fotc_traffic_control_and_status
fotc_transfer_data
link_details
long_term_data
other_status_for_highways
other_status_for_roads
planned_events_local_data
request_other_current_incidents_data
request_other_planned_events_data
request_other_TMC_data

Output Flows

other_control_data_for_highways
other_control_data_for_roads
other_current_incidents
other_planned_events
other_TMC_cv_incidents
other_TMC_emergency_data
other_TMC_strategy_data
other_traffic_center_data
request_local_current_incidents_data
request_local_planned_events_data
status_data_for_highways
status_data_for_roads
totc_data_request
totc_identity
totc_traffic_control_and_status
totc_transfer_data

Description:

Overview: This process shall exchange data with similar processes in other Traffic Management Subsystems (TMS's). The other TMS can be adjacent geographically, under control of a different jurisdiction, or part of a more complex hierarchy. The exchange of data shall be triggered by either a request from a remote TMS for data from the TMS to which the Manage Traffic function belongs, or because data needs to be sent from the local TMS to a remote TMS. This data shall include traffic control preemption for vehicle routes which
pass through the local network but have a destination in an area served by a remote TMS, or include data about an incident that has an impact on the traffic conditions in the network served by a remote TMS. The data received from remote TMS's shall be used either to vary the current traffic control strategy to give signal preemption to emergency vehicles or enable the passage of commercial vehicles with unusual loads, or as input to the local traffic predictive model estimation process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_incidents_for_other_TMC';
(b) 'emergency_data_for_other_TMC';
(c) 'fotc-data_request';
(d) 'fotc-identity';
(e) 'fotc-transfer_data';
(f) 'request_other_TMC_data';
(g) 'request_other_current_incidents_data';
(h) 'request_other_predicted_incidents_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'link_details';
(b) 'historical_data';
(c) 'current_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from the other TM terminator:
(a) 'fotc-identity';
(b) 'fotc-transfer_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'other_current_incidents';
(b) 'other_planned_events';
(c) 'other_TMC_cv_incidents';
(d) 'other_TMC_emergency_data';
(e) 'other_TMC_strategy_data';
(f) 'other_traffic_center_data';
(g) 'request_local_current_incidents_data';
(h) 'request_local_planned_events_data';
(i) 'totc-data_request';
(j) 'totc-identity';
(k) 'totc-transfer_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either 'cv_incidents_for_other_TMC' or 'emergency_data_for_other_TMC' unsolicited inputs are received, generate the 'totc-identity' and 'totc-transfer_data' solicited output data flows (and issue them to the other TM terminator);
(c) when the 'fotc-data_request' and 'fotc-identity' unsolicited input data flows are received, and if requested in those data flows, read the data from the long term data store that is relevant to the requesting TMS, and generate the 'totc-identity' and 'totc-transfer_data' solicited output data flows, (and issue them to the other TM terminator);
(d) when the 'fotc-identity' and 'fotc-transfer_data' unsolicited input data flows are received, generate those of the 'other_current_incidents', 'other_planned_events', 'other_TMC_cv_incidents' or 'other_TMC_emergency_data' solicited output data flows for which data has been provided and send them to their receiving processes, or in the case of the 'other_TMC_emergency_data' flow load the data into the store containing other traffic center data;
(e) when any of the 'request_other_TMC_data', 'request_other_current_incidents_data' or 'request_other_planned_events_data' unsolicited input data flows is received, generate the 'totc-data_request' and 'totc-identity' solicited output data flows and send them to the other TM terminator;
(f) the process shall be responsible for the maintenance of the store of data from other TMC's for use by the predictive modeling process.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.2.5;
USR = 1.6.2.5.2;
USR = 1.6.3;
USR = 1.6.3.6;
USR = 1.6.4;
USR = 1.6.4(a);
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(d);

Output Flow Dynamics Assumptions:

other_current_incidents = fotc-transfer_data;
other_planned_events = fotc-transfer_data;
other_TMC_cv_incidents = fotc-transfer_data;
other_TMC_emergency_data = fotc-transfer_data;
other_TMC_strategy_data = fotc-transfer_data;

other_traffic_center_data = fotc-transfer_data;
other_control_data_for_highways = fotc-traffic_data;
other_control_data_for_roads = fotc-traffic_data;
other_status_for_roads = fotc-traffic_data;
other_status_for_highways = fotc-traffic_data;
request_local_current_incidents_data = fotc-transfer_data;
request_local_planned_events_data = fotc-transfer_data;
totc-data_request = request_other_TMC_data;
totc-identity = request_other_TMC_data;
totc-transfer_data = fotc-data_request;
totc-traffic_control_and_status = fotc-traffic_control_and_status;
status_data_for_highways = totc-traffic_control_and_status;
status_data_for_roads = totc-traffic_control_and_status;

1.1.6 Collect Vehicle Tag Data for Link Time Calculations

Input Flows

parking_lot_tag_data_input
toll_tag_data_input

Output Flows

parking_lot_tag_data_needed
toll_tag_data_needed
vehicle_tag_data

Description:

Overview: This process shall collect data from toll and parking tags on passing vehicles. This shall be achieved by transmitting a tag data request message and collecting any tag reply data that is received. This reply data shall be translated by the process into a unique but anonymous ID that does not store or transmit the identity of the tag in any way that is traceable to the tag owner, e.g. credit identity or stored credit value. This ID is then passed on to another process for further link travel time calculation analysis.

Unsolicited Output Processing: This process shall provide the following output flows regardless of whether or not it has received any input flows:
(a) 'parking_lot_tag_data_needed';
(b) 'toll_tag_data_needed'.

Solicited Input Processing: This process shall receive the following data flows as a result of data being sent to other processes:
(a) 'parking_lot_tag_data_input';
(b) 'toll_tag_data_input'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'vehicle_tag_data'.

Functional Requirements: This process shall:
(a) continuously output the unsolicited output flows list above;
(b) when either of the solicited input flows shown above are received, convert the tag data into a form which is unique and protects the identity of the traveler;
(c) output the data obtained in (b) to the analysis process in the 'vehicle_tag_data' data flow.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.2;
USR = 1.6.2.2;
USR = 1.6.2.4;
USR = 1.6.2.4.1;
USR = 1.6.2.5.1;

Output Flow Dynamics Assumptions:

parking_lot_tag_data_needed = 1;
toll_tag_data_needed = 1;
vehicle_tag_data = 1;

1.1.7 Collect Vehicle Smart Probe Data

Input Flows

vehicle_smart_probe_data

Output Flows

vehicle_smart_probe_input_data

Description:

Overview: This process shall collect data from vehicle smart probes. This data shall include information about conditions in the vicinity of the vehicle operating as a smart probe. It shall receive this data from passing vehicles and shall add its own identity and location before sending the data on to the process which outputs the data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'vehicle_smart_probe_data'.

Solicited Output Processing: This process shall provide the following output flow as a result of the above input being received:
(a) 'vehicle_smart_probe_input_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flow listed above;
(b) when flow in (a) is received, add the process' location and identity (for this instance of the process) to the data;
(c) output the combined data produced in (b) in the vehicle smart probe input data message to the analysis process.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.2;
USR = 1.6.2.2;
USR = 1.6.2.3;
USR = 1.6.2.3.1;
USR = 1.6.2.3.2;
USR = 1.6.2.4;
USR = 1.6.2.5;

Output Flow Dynamics Assumptions:

vehicle_smart_probe_input_data = SMART_PROBE_RATE;

1.2.1 Select Strategy

Input Flows

current_road_network_use
cv_incident_override
demand_overrides
emergency_traffic_control_request
ftop_strategy_override
ftop_video_camera_strategy_change
incident_strategy_override
indicator_fault_state
indicator_input_state_for_highways
indicator_input_state_for_roads
other_TMC_cv_incidents
other_TMC_emergency_data
other_TMC_strategy_data
special_vehicle_priority_routing
static_data_for_strategy

Output Flows

cv_incidents_for_other_TMC
emergency_data_for_other_TMC
emergency_traffic_control_response
request_other_TMC_data
selected_highway_control_strategy
selected_hri_control_strategy
selected_parking_lot_control_strategy
selected_ramp_control_strategy
selected_road_control_strategy
selected_strategy
video_camera_control_strategy

Description:

Overview: This process shall select the appropriate traffic control strategy to be implemented over a road and/or freeway section served by the specific instance of the Manage Traffic function. The strategy shall be selected by the process from a number that are available, e.g. adaptive control, fixed time control, local operations. The selected strategy shall be passed by the process to the actual control processes for implementation according to the part of the network to which it is to be applied, i.e. surface roads, freeways (i.e. limited access roads), ramps and/or parking lots. The definition of strategy can be extended to include a strategy for the operations of sensors such as video cameras used to provide traffic surveillance data. The process shall make it possible for the current strategy selection to be modified to accommodate the effects of such things as incidents, emergency vehicle preemption, the passage of commercial vehicles with unusual loads, equipment faults

and overrides from the traffic operations personnel. The strategy for control of freeways and parking lots is through use of DMS signs and lane indicators. The strategy for control of ramps is through the timing plans for ramp meters. The selected strategy shall be sent to the process within the Provide Traffic Surveillance facility responsible for maintaining the store of long term data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'current_road_network_use';
(b) 'cv_incident_override';
(c) 'demand_overrides';
(d) 'emergency_traffic_control_request';
(e) 'ftop-strategy_override';
(f) 'ftop-video_camera_strategy_change';
(g) 'incident_strategy_override';
(h) 'indicator_fault_state';
(i) 'indicator_input_state_for_highways';
(j) 'indicator_input_state_for_roads';
(k) 'special_vehicle_priority_routing';
(l) 'other_TMC_cv_incidents';
(m) 'other_TMC_emergency_data';
(n) 'other_TMC_strategy_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'static_data_for_strategy'.

Unsolicited Output Processing: This process shall provide the following output flows regardless of any inputs that are received:
(a) 'request_other_TMC_data';
(b) 'video_camera_control_strategy'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_incidents_for_other_TMC';
(b) 'emergency_data_for_other_TMC';
(c) 'selected_parking_lot_control_strategy';
(d) 'selected_ramp_control_strategy';
(e) 'selected_highway_control_strategy';
(f) 'selected_road_control_strategy';
(g) 'selected_strategy';
(h) 'emergency_traffic_control_response'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) determine the traffic control strategy that provides the best possible traffic conditions within the road network served by the Manage Traffic function. The definition of 'best' shall be a local policy decision. (An example might be a strategy that minimizes stops and delays thus reducing 'stop-start' travel and fuel consumption and the environmental impact of travel.);
(c) in determining the strategy, the process shall be able to use data provided as input from other parts of the Manage Traffic function, unless countermanded by input from the traffic operations personnel, or the default strategy in the store of static data, as well as the fault state of all indicator equipment;
(d) if no input is available from other parts of the Manage Traffic function, then the strategy defined in the store of static data shall be used;
(e) if in (d) no strategy is specified, the process shall allow all controlled equipment to operate under local control, setting all variable message sign (dms) outputs to 'blank face' indicating that there is no message;
(f) where the inputs from other parts of the Manage Traffic function lead to conflicts in the required strategy to be selected, the process shall observe a locally determined order of priority. For example, the following order of priority might be followed: emergency vehicle route, incident strategy override, multimodal crossing inputs, operator strategy override, demand strategy override, low traffic volume route, commercial vehicle route, analysis of the road network use and background strategy selection from the store of static data;
(g) data for emergency and commercial vehicle routes sent from other Traffic Management Subsystems (TMS's) shall be given the same level of importance as those that originate locally (unless locally overridden);
(h) the process shall automatically cancel strategies selected by traffic operations personnel at a locally determined time and/or period after they were imposed, if they have not been canceled previously, to avoid unintended effects on the traffic control strategies for other days;
(i) when a new strategy has been determined, it shall be sent to other processes in the Manage Traffic function for implementation;
(j) the output in (i) shall only be sent to those processes that serve equipment specified in the new strategy;
(k) changes in the current strategy must always be immediately sent to another part of the Manage Traffic function for loading into the long term data store.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.3;
USR = 1.6.3.5;
USR = 1.6.3.6;

Output Flow Dynamics Assumptions:

cv_incidents_for_other_TMC = cv_incident_override;
emergency_data_for_other_TMC = emergency_vehicle_route;
request_other_TMC_data = 12/(60*60);
selected_highway_control_strategy = 12/60;
selected_road_control_strategy = 12/60;
selected_hri_control_strategy = 12/60;
selected_ramp_control_strategy = 12/60;
selected_parking_lot_control_strategy = 4/60;

selected_strategy = 2*(12/60)+4/60;
traffic_video_camera_control = ftop-video_camera_action_request;
emergency_traffic_control_response = emergency_traffic_control_request;
video_camera_control_strategy = 12/60;

1.2.2.1 Determine Indicator State for Freeway Management

Input Flows

coordination_data_roads_to_freeways
prediction_data
selected_highway_control_strategy
static_data_for_highways
strategy_data_for_highways
transit_highway_overall_priority

Output Flows

coordination_data_freeways_to_roads
current_highway_network_data
current_highway_network_state
indicator_highway_requested_state
transit_highway_priority_given

Description:

Overview: This process shall implement selected traffic control strategies and transit vehicle overall priority on some or all of the indicators covering the freeway network served by the Manage Traffic function. It shall implement the strategies only using the indicators (e.g. dynamic message signs (DMS)) specified in the implementation request and shall coordinate its actions with those of the process that controls the road network. The process shall also be capable of monitoring the extra inputs that will arise where tunnels are involved, including the detection of fire and the consequent requirement to re-route traffic.

Data Flows: All input data flows are unsolicited and all output data flows are solicited with the exception of the following:
(a)'static_data_for_highways', which is data accessed from a local data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) immediately implement any strategy requests only using the indicators specified in the request;
(c) it shall be possible for the strategy request to require implementation on one, some or all the indicators that are available (and not faulty) in the freeway network served by the Manage Traffic function;
(d) strategy implementation must make use of the freeway sign sequences to ensure that signs are set in a manner that is safe for all freeway users;
(e) requests for high occupancy vehicle (hov) and transit priority shall be executed immediately but not take precedence over emergency vehicle routes;
(f) special consideration must be given to conditions in tunnels and in particular to the need to automatically implement alternative traffic management strategies to route traffic away from fires or similar extreme hazards that may be detected;
(g) the process shall use the strategy data input to monitor the effects of the currently selected strategies and make small adjustments which will further improve the efficiency of the current traffic flow;
(h) transit priority shall be implemented on the indicators covering the requested route(s) and its confirmation of its implementation shall be sent back to the requesting process in the Manage Transit function;
(i) the process shall implement any changes in control in a safe manner that does not in any way endanger vehicles and/or their drivers, pedestrians or operators of non-motorized vehicles;
(j) send each change in strategy to another process in the Manage Traffic function for loading into the store of long term data;
(k) send the required indicator state to another process in the Manage Traffic function for output to the roadside equipment that drives the indicators.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.1;

Output Flow Dynamics Assumptions:

coordination_data_freeways_to_roads = selected_highway_control_strategy;
current_highway_network_data = selected_highway_control_strategy;
current_highway_network_state = selected_highway_control_strategy;
indicator_highway_requested_state = selected_highway_control_strategy;
transit_highway_priority_given = transit_highway_overall_priority;

1.2.2.2 Determine Indicator State for Road Management

Input Flows

coordination_data_freeways_to_roads
coordination_data_ramps_to_roads
prediction_data
selected_hri_control_strategy
selected_road_control_strategy
static_data_for_roads
strategy_data_for_roads
transit_road_overall_priority

Output Flows

coordination_data_roads_to_freeways
coordination_data_roads_to_ramps
current_road_network_data
current_road_network_state
indicator_road_requested_state
transit_road_priority_given

Description:

Overview: This process shall implement selected traffic control strategies and
transit priority on some or all of the indicators covering the road (surface street) network served
by the Manage Traffic function. It shall implement the strategies only using the indicators
(intersection and pedestrian controllers, variable message signs (dms), etc.) that are specified in
the implementation request and shall coordinate its actions with those of the processes that control
the freeway network and the ramps that give access to the freeway network.

Data Flows: All input data flows are unsolicited and all output data flows are solicited with the
exception of the following:
(a)'static_data_for_roads', which is data received from a local data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) immediately implement any strategy requests only using the indicators specified in the request;
(c) control all indicators that are intersection and
pedestrian controllers using a methodology which responds to vehicles and pedestrians in a locally
determined manner;
(d) where vehicle and pedestrian responsive control cannot be implemented, or is not specified in
the strategy request, the following traffic control methodologies shall be available to the process
for implementation within some or all of the controlled network: fixed time control sequences
(usually referred to as fixed time plans), the automatic selection of the most appropriate fixed
time plan on the basis of current real time traffic data, the selection of special fixed time plans
to cover such things as bridges opening when requested by the specific data input and the ability
of one or more device(s) to operate under its own (local) control;
(e) the process must be capable of implementing the required control strategy on one, some or all
the indicators that are available (and not faulty) in the road network;
(f) traffic control preemption shall be capable of being implemented for emergency or special priority
vehicles.
(g) requests for high occupancy vehicle (hov) and transit priority shall be executed immediately
but not take precedence over emergency vehicle routes;
(h) the process shall use the strategy data input to monitor the effects of the currently selected
strategies and make any small adjustments which will further improve the efficiency of the traffic flow;
(i) transit priority shall be implemented on the indicators covering the requested route(s) and
confirmation of its implementation shall be sent back to the requesting process in the Manage Transit
function;
(j) the process shall implement any changes in control in a safe manner that does not in any way endanger vehicles and/or
their drivers, pedestrians or operators of non-motorized vehicles;
(k) send each change in strategy to another process in the Manage Traffic function for loading into
the store of long term data;
(l) send the required indicator state to another process in the Manage Traffic function for output
to the roadside equipment that drives the indicators.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.1;

USR = 1.6.1.1;
USR = 1.6.1.1.1;
USR = 1.6.1.1.2;
USR = 1.6.1.1.3;
USR = 1.6.1.1.4;
USR = 1.6.1.1.5;
USR = 1.6.1.2;
USR = 1.6.1.2.1;
USR = 1.6.1.2.2;
USR = 1.6.1.2.3;
USR = 1.6.1.3;
USR = 1.6.1.4;
USR = 1.6.1.4.1;
USR = 1.6.1.5;
USR = 1.6.1.6;
USR = 1.6.1.7;
USR = 1.6.1.7(b);
USR = 1.6.3;
USR = 1.6.3.1;
USR = 1.6.3.2;
USR = 1.6.3.2.1;
USR = 1.6.3.2.2;
USR = 1.6.3.3;
USR = 1.6.3.3.1;
USR = 1.6.3.3.2;
USR = 1.6.3.3.3;
USR = 1.6.3.3.4;
USR = 5.0;
USR = 5.2;
USR = 5.2.3;
USR = 5.2.3.1;
USR = 5.2.3.2;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(a);

Output Flow Dynamics Assumptions:

coordination_data_roads_to_freeways = selected_road_control_strategy;
coordination_data_roads_to_ramps = 12/(60*60);
current_road_network_data = selected_road_control_strategy;
current_road_network_state = selected_road_control_strategy;
indicator_road_requested_state = selected_road_control_strategy + selected_hri_control_strategy;
transit_road_priority_given = transit_road_overall_priority;

1.2.3 Determine Ramp State

Input Flows

coordination_data_roads_to_ramps
ramp_data
selected_ramp_control_strategy
static_data_for_ramps
transit_ramp_overall_priority

Output Flows

coordination_data_ramps_to_roads
current_ramp_state
ramp_signal_state
transit_ramp_priority_given

Description:

Overview: This process shall implement the selected control strategies on some or all of the freeway entry ramps in the freeway network served by the Manage Traffic function. It shall implement the strategies only using the ramps that are specified in the implementation request and shall coordinate its actions with those of the process that controls the road network. The process shall base its ramp metering decisions on the data from sensors and ramp meters monitoring traffic conditions upstream and downstream of the ramps. Data from sensors on the ramp used to detect flow past the meter, extent of queues on the ramp, and the presence of vehicles will also be used as the basis for the ramp metering decisions. The decision making process shall use an algorithm to determine the ramp's state based on the ramp control strategy and the sensor input data received. The process shall coordinate its activities with the process responsible for controlling the road(surface street) network.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a)'coordination_data_roads_to_ramps';
(b)'ramp_data';
(c)'selected_ramp_control_strategy';
(d)'transit_ramp_overall_priority';
(e)'transit_ramp_preemptions'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a)'static_data_for_ramps'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a)'coordination_data_ramps_to_roads';
(b)'current_ramp_state';
(c)'ramp_signal_state';
(d)'transit_ramp_priority_given'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) in its control of the ramp metering equipment, the process shall use an algorithm which enables traffic to be admitted to a freeway only if it can be absorbed into the existing flow without causing serious disruption, i.e. the existing flow is not at a level which produces the most efficient traffic flow;
(c) the inputs to the algorithm in (b) shall be obtained by processing traffic flow data received from sensors which monitor freeway conditions both upstream and downstream of the ramp;
(d) the process shall be able to override the requirements in (b) and open the ramp if coordination data received from the process controlling the roads (surface streets) indicates that closing it will have a greater negative impact on the road (surface street) network that is upstream of the ramp;
(e) be able to give priority to high occupancy vehicles (hov) and transit vehicles, particularly when priority requests are received for the latter;
(f)allow transit priority to be implemented on the indicators covering the requested route(s) and confirmation of its implementation shall be sent back to the requesting process in the Manage Transit function;
(g) be capable of implementing the required control strategy on one, some or all the ramps that are available (and not faulty) in the freeway network served by the function;
(h) send each change in strategy to another process in the Manage Traffic function for loading into

the store of long term data;
(i) output coordination data to the process controlling the roads (surface streets).

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.1;
USR = 1.6.1.1;
USR = 1.6.1.1.2;
USR = 1.6.1.2;
USR = 1.6.1.2.1;
USR = 1.6.1.2.3;

Output Flow Dynamics Assumptions:

coordination_data_ramps_to_roads = 12/(60*60);
current_ramp_state = 1;
ramp_signal_state = 1;
transit_ramp_priority_given = transit_ramp_overall_priority;

1.2.4.1 Output Control Data for Roads

Input Flows

dms_status_for_roads
dms_updates_for_roads
har_status_for_roads
hri_guidance_for_dms
indicator_input_data_from_roads
indicator_road_requested_state
other_control_data_for_roads
parking_guidance_for_dms
static_data_for_road_control
status_data_for_roads
vehicle_pollution_message_for_roads

Output Flows

control_data_for_roads
dms_data_for_roads
har_data_for_roads
indicator_control_data_for_roads
indicator_control_monitoring_data_for_roads
indicator_control_storage_data_for_roads
indicator_data_fault_for_roads
indicator_input_state_for_roads
indicator_input_storage_data_for_roads
indicator_sign_control_data_for_hri
other_status_for_roads
vehicle_sign_data_for_roads

Description:

Overview: This process shall transfer data to processes responsible for controlling equipment located at the roadside within the road (surface street) network served by the Manage Traffic function. This data shall contain outputs for use by roadside indicators, such as intersection and pedestrian controllers, dynamic message signs (DMS), highway advisory radio (HAR), etc. Data for use by in-vehicle signage equipment shall be sent to another process for output to roadside processes.

All data shall be sent to this process by processes within the Manage Traffic function. This process shall also be responsible for the monitoring of input data showing the way in which the indicators

are responding to the data that they are being sent, and the reporting of any errors in their responses as faults to the Collect and Process Indicator Fault Data facility within the Manage Traffic function. All output and input data shall be sent by the process to another process in the Manage Traffic function to be loaded into the store of long term data.

Data Flows: All input data flows are unsolicited with the exception of static_data_for_control which is solicited along with all output data flows.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any change occurs to the input data, change the appropriate indicator output data;
(c) as a result of (b), update the vehicle signage data, adding the location and identity of the route segments from which the indicator data can be seen from the static data for dms allocation;
(d) maintain communication with all indicators so that they will continue to obey the data contained in the data that is being sent to them;
(e) immediately report all indicators that fail to respond to the commands in the data that they have been sent to the processes responsible for fault management.

User Service Requirements:

USR = 1.0;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(a);
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.1.2.1;

USR = 1.6.1.4;
USR = 1.6.3;
USR = 1.6.3.3;
USR = 1.6.3.3.1;
USR = 1.6.3.3.2;
USR = 1.6.3.3.3;
USR = 1.6.3.3.4;
USR = 1.6.3.4;
USR = 1.6.3.4(a);
USR = 1.6.3.4.1;
USR = 1.10;

Output Flow Dynamics Assumptions:

indicator_control_data_for_roads = 1;
indicator_control_monitoring_data_for_roads = 1;
indicator_control_storage_data_for_roads = 1;
indicator_data_fault_for_roads =
(INTERSECTIONS+PEDESTRIANS+CROSSINGS+GRADE_CROSSINGS+SIGNS+RAMPS)*1/(60*60*24*7*52);
indicator_input_state_for_roads = 1;
indicator_input_storage_data_for_roads = 1;
har_data_for_roads = 1/60;
vehicle_sign_data_for_roads = 1;
indicator_sign_control_data_for_hri = 1;
control_data_for_roads = 1;
other_status_for_roads = 1;
dms_data_for_roads = 1;

1.2.4.2 Output Control Data for Freeways

Input Flows

dms_status_for_highways
dms_updates_for_highways
har_status_for_highways
indicator_highway_requested_state
indicator_input_data_from_highways
other_control_data_for_highways
parking_guidance_for_dms
ramp_signal_state
static_data_for_highway_control
status_data_for_highways
vehicle_pollution_message_for_highways

Output Flows

control_data_for_highways
dms_data_for_highways
har_data_for_highways
indicator_control_data_for_highways
indicator_control_monitoring_data_for_highways
indicator_control_storage_data_for_highways
indicator_data_fault_for_highways
indicator_input_state_for_highways
indicator_input_storage_data_for_highways
other_status_for_highways
vehicle_sign_data_for_highways

Description:

Overview: This process shall transfer data to processes responsible for controlling equipment located at the roadside within the freeway network served by the Manage Traffic function. This data shall contain outputs for use by roadside indicators, such as dynamic message signs (DMS), etc. Data for use by in-vehicle signage equipment shall be sent to another process for output to roadside processes. All data shall have been sent to this process by processes within the Manage Traffic function. This process shall also be responsible for the monitoring of input data showing the way in which the indicators are responding to the data that they are being sent, and the reporting of any errors in their responses as faults to the Collect and Process Indicator Fault Data facility within the Manage Traffic function. All output and input data shall be sent by the process to another process in the Manage Traffic function to be loaded into the store of long term data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'indicator_input_data';
(b) 'indicator_highway_requested_state';
(c) 'indicator_road_requested_state';
(d) 'ramp_signal_state';
(e) 'vehicle_pollution_message';
(f) 'dms_parking_guidance';
(g) 'dms_updates'
(h) 'har_status_for_highways'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'static_data_for_control'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_control_data';
(b) 'indicator_control_monitoring_data';
(c) 'indicator_control_storage_data';
(d) 'indicator_data_fault';
(e) 'indicator_input_state';
(f) 'indicator_input_storage_data';
(g) 'vehicle_sign_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any change occurs to the input data, change the appropriate indicator output data;

(c) as a result of (b), update the vehicle signage data, adding the location and identity of the route segments from which the indicator data can be seen from the static data for dms allocation;
(d) maintain communication with all indicators so that they will continue to obey the data contained in the data that is being sent to them;
(e) immediately report all indicators that fail to respond to the commands in the data that they have been sent to the processes responsible for fault management.

User Service Requirements:

USR = 1.0;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(a);
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;

Output Flow Dynamics Assumptions:

indicator_control_data_for_highways = 1;
indicator_control_monitoring_data_for_highways = 1;
indicator_control_storage_data_for_highways = 1;
indicator_data_fault_for_highways = (CROSSINGS+SIGNS+RAMPS)*1/(60*60*24*7*52);
indicator_input_state_for_highways = 1;
indicator_input_storage_data_for_highways = 1;
vehicle_sign_data_for_highways = 1;
other_status_data_for_highways = 1;
dms_data_for_highways = 1;
control_data_for_highways = 1;
har_data_for_highways = 1;

1.2.4.3 Output In-vehicle Signage Data

Input Flows

current_incident_data_for_vehicle_signage
hri_guidance_for_beacon_message
planned_event_data_for_vehicle_signage
static_data_for_vehicle_signage
traffic_data_for_signage
vehicle_sign_data_for_highways
vehicle_sign_data_for_roads
vehicle_signage_incident_data

Output Flows

vehicle_sign_data
vehicle_signage_incident_data

Description:

Overview: This process shall format and output data for use by roadside processes in creating in-vehicle signage. This process supports a full range of functionality for in-vehicle signage (from display of signage to location specific advisory data). The process shall be capable of outputting some or all of the following advisory data: link state data, current incidents, planned events, and highway rail intersection status. The process shall be capable of outputting some or all of the following signage data: dynamic message sign contents or fixed signage. The data shall be structured by this process so that it can be output by each roadside process to vehicles for use by in-vehicle signage equipment.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a)'current_incident_data_for_vehicle_signage';
(b)'planned_event_data_for_vehicle_signage';
(c)'static_data_for_vehicle_signage';
(d)'traffic_data_for_signage';
(e)'vehicle_sign_data_for_highways';
(f)'vehicle_sign_data_for_roads'
(g)'hri_guidance_for_beacon_message'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a)'vehicle_signage_incident_data'.

Solicited Output Processing: This process shall provide the following output flow as a result of the above inputs being received:
(a)'vehicle_sign_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any change occurs in either of 'current_incident_data_for_vehicle_signage', 'planned_event_data_for_vehicle_signage', 'traffic_data_for_signage', 'vehicle_sign_data_for_highways', 'vehicle_sign_data_for_roads' or 'hri_guidance_for_beacon_message' data flows, then change the contents of the signage output data flow to be the same;
(c) as a result of (b), output the vehicle signage data, filtering the incident data so that the roadside processes only receive data which is relevant to their location, and only providing the sign data that relates to the signs covered by each roadside process;
(d) the process shall perform the incident location filtering using the roadside process location data in the 'static_data_for_vehicle_signage' data flow;
(e) when new incident data is received in either of 'current_incident_data_for_vehicle_signage' or planned_event_data_for_vehicle_signage', load it into the store of incident data, updating any previously received data that relates to the same incident.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.1.2.1;
USR = 1.6.1.4;
USR = 1.6.3;

USR = 1.6.3.3;
USR = 1.6.3.3.1;
USR = 1.6.3.3.2;
USR = 1.6.3.3.3;
USR = 1.6.3.3.4;
USR = 1.6.3.4;
USR = 1.6.3.4.1;
USR = 1.10;
USR = 1.10.0;
USR = 1.10.1;
USR = 1.10.1.1;

<u>Output Flow Dynamics Assumptions:</u>

vehicle_sign_data = vehicle_sign_data_for_highways+current_incident_data_for_vehicle_signage
+vehicle_sign_data_for_roads+planned_event_data_for_vehicle_signage;
vehicle_signage_incident_data = vehicle_sign_data_for_highways+vehicle_sign_data_for_roads
+current_incident_data_for_vehicle_signage
+planned_event_data_for_vehicle_signage;

1.2.5.1 Determine Parking Lot State

Input Flows

parking_input_data
parking_lot_calculated_occupancy
parking_lot_operator_input_data
selected_parking_lot_control_strategy

Output Flows

parking_guidance_for_dms
parking_lot_current_state
parking_lot_occupancy
parking_lot_operator_output_data
parking_lot_state_for_archive
parking_output_data

Description:

Overview: This process shall implement the selected control strategies on some or all of the parking lots in the surface street and freeway network served by the Manage Traffic function. It shall use the current parking lot occupancy provided by another process to determine the parking lot state to be used in sign settings implemented by other processes in the function, when this is not subject to a strategy override. The parking lot state shall be determined using threshold occupancy values contained in the static data provided by the Plan System Deployment function. Fixed thresholds for the states: 'full', 'almost full' and 'available' are in the data flow 'static_data_for_parking_lots'. In addition, threshold transitions can depend on whether the actual occupancy of the lot is increasing or decreasing, allowing hysteresis in the parking lot state transitions, so as to control jitter between parking lot states. The process shall also provide the current parking lot occupancy to other processes in the function.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'parking_lot_calculated_occupancy';
(b) 'parking_lot_operator_input_data';
(c) 'parking_input_data';
(d) 'selected_parking_lot_control_strategy';
(e) 'static_data_for_parking_lots'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'parking_lot_current_state';
(b) 'parking_lot_occupancy';
(c) 'parking_lot_operator_output_data';
(d) 'parking_output_data';
(e) 'dms_parking_guidance_for_highways';
(f) 'dms_parking_guidance_for_roads'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) calculate the current parking lot state by comparing the occupancy provided from one or more other processes with the threshold values contained in the static data provided by the unsolicited input flow 'static_data_for_parking_lots';
(c) if the current parking lot occupancy is available from more than one source, then use the calculated value as first priority;
(d) if a requested state is received from the strategy selection process, then use it to over-write the calculated parking lot state;
(e) use the parking lot state to determine the dms state needed to advise motorists of the spaces available in each lot or to guide them to lots which have spaces available, sending the data out through the data flow for dynamic messages signs (dms) parking_guidance_for_dms;
(f) send each change in state to another process in the Manage Traffic function for loading into the store of long term data;
(g) output the unsolicited data flow shown above to the Plan System Deployment function.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.2;
USR = 1.8.2.1;

USR = 1.8.2.1(c);
USR = 1.8.3;
USR = 1.8.3.1;

Output Flow Dynamics Assumptions:

parking_lot_current_occupancy = 1;
parking_lot_current_state = 1;
parking_lot_occupancy = 12/(60*60);
parking_lot_operator_output_data = 12/(60*60);
parking_output_data = 12/(60*60);
parking_guidance_for_dms = 1;
parking_lot_state_for_archive = 1/(60*60);

1.2.5.2 Coordinate Other Parking Data

Input Flows

fop_parking_coordination_data
other_parking_lot_price_data
parking_lot_availability
parking_output_data
parking_transit_update

Output Flows

other_parking_lot_price_data_request
parking_input_data
top_parking_coordination_data

Description:

Overview: This process shall process shall continuously communicate and exchange data with parking operators and systems. The exchange of data shall be triggered by either a request from a remote Parking Management Subsystem for data from the operators or systems the Provide Electronic Payment function belongs, or because data needs to be sent from the local Parking Management Substystem to another remote Parking Management Subsystem. This data shall include parking lot state, parking price information, parking availability, etc.

Data Flows: All inputs are unsolicited and all outputs are solicited with the exception of parking_input_data which is generated as an unsolicited output.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) always attempt to provide output of the parking coordination data regardless of the inputs that are (or are not) received from the system;
(c) respond to any requests for price data from other parking systems by forwarding the request for data to the Provide Electronic Payment function.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(c);

Output Flow Dynamics Assumptions:

parking_input_data = 1;
other_parking_lot_price_data_request = 12/(60*60);
top-parking_coordination_data = 12/(60*60);

1.2.5.3 Provide Parking Lot Operator Interface

Input Flows

fpo_current_lot_state
fpo_lot_occupancy
parking_lot_operator_output_data
parking_lot_operator_transit_update

Output Flows

parking_lot_operator_input_data
tpo_change_lot_state

Description:

Overview: This process shall provide the interface to a local parking lot operator that controls the use of the lot. The operator shall provide inputs of occupancy and/or the current lot state to this process. This process shall provide the operator with outputs that request a change to the lot state, which the operator shall implement by activating local dynamic message signs (DMS) and controlling the use of entry/exit barriers, and data about transit services that provide a park and ride (P+R) operation to be output through local DMS.

(a) 'fpo-current_lot_state';
(b) 'fpo-lot_occupancy';
(c) 'parking_lot_operator_output_data';
(d) 'parking_lot_operator_transit_update'.

Unsolicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'parking_lot_operator_input_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tpo-change_lot_state'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) always attempt to provide output of the parking lots current state regardless of the inputs that are (or are not) received from the operator;
(c) respond to any requests for changes in the parking lot state received from other processes in the Manage Traffic function, regardless of any local inputs.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

parking_lot_operator_input_data = 1;
tpo-change_lot_state = 12/(60*60);

1.2.5.4 Determine P+R needs for Transit Management

Input Flows

parking_lot_occupancy
parking_lot_transit_response

Output Flows

parking_lot_operator_transit_update
parking_lot_transit_request
parking_transit_update

Description:

Overview: This process shall be responsible for calculating the need for transit services to provide a park and ride (P+R) operation at a parking lot. This calculation shall be based on the rate of change of the current parking lot occupancy. The results of the calculation shall be sent to the Manage Transit function in the form of a request for an additional (or reduced) level of service, depending on demand at the parking lot. The results of the request shall also be passed to other processes within the function.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'parking_lot_occupancy'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'parking_lot_transit_response'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'parking_lot_operator_transit_update';
(b) 'parking_transit_update';
(c) 'parking_lot_transit_request'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) calculate the need for transit services based on the rate at which vehicles are arriving at the parking lot and pass this information on to the Manage Transit function;
(c) communicate the response from the Manage Transit function to the processes responsible for monitoring the operation of the parking lot.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(a);

Output Flow Dynamics Assumptions:

parking_lot_operator_transit_update = 20/(60*60);
parking_lot_transit_request = 20/(60*60);
parking_transit_update = 20/(60*60);

1.2.5.5 Manage Parking Archive Data

Input Flows

fpo_archive_commands
parking_archive_request
parking_archive_status
parking_charge_response_for_archive
parking_data_archive
parking_lot_state_for_archive

Output Flows

parking_archive_data
parking_charge_request_for_archive
parking_data_archive
tpo_archive_status

Description:

Overview: This process shall obtain parking lot availability and charge data and distribute it to the Manage Archive Data function. The process shall run when a request for data is received from an external source.

Data Flows: All input data flows from the are unsolicited with the exception of parking_archive_status and all output flows which are solicited.

Functional Requirements: This process shall meet the following functional requirements;
(a) continuously monitor for receipt for the unsolicited input flows listed above;
(b) when any of the unsolicited inputs shown above are received, the process shall immediately generate the solicited output shown above;
(c) data shall only be sent to the source from which the data request originated.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(e);
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(e);
USR = 7.1.3.1.10;

Output Flow Dynamics Assumptions:

parking_archive_data = parking_archive_data_request;
parking_charge_request_for_archive = parking_charge_response_for_archive;
parking_data_archive = parking_data_archive;
tpo-archive_status = fpo-archive_commands;

1.2.5.6 Calculate Parking Lot Occupancy

Input Flows

parking_lot_input_data
static_data_for_parking_lots

Output Flows

parking_lot_calculated_occupancy

Description:

Overview: This process shall calculate the occupancy of a parking lot based on processed traffic sensor data provided by other processes within the Manage Traffic function. The process shall use the static data for parking lots to determine the part(s) of the supplied data that apply to its entry and exit lanes, so that the numbers of vehicles entering and leaving can be calculated. These calculated flows shall be used by the process to generate the current parking lot occupancy.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'parking_lot_input_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'static_data_for_parking_lots'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'parking_lot_calculated_occupancy'.

Functional Requirements: This process shall:
(a) continuously monitor the unsolicited input 'parking_lot_input_data';
(b) use the static data provided as the solicited input flow shown above to determine which part(s) of the data provided by the data flow in (a) apply to the parking lot's entry and exit lanes;
(c) use the data identified in (b) to calculate the numbers of vehicles entering and leaving the parking lot and hence calculate the parking lot occupancy;
(d) output the data obtained in (c) as the 'parking_lot_calculated_occupancy' solicited output flow.

User Service Requirements:

USR = 1.0;
USR = 1.1;
USR = 1.1.0;
USR = 1.1.1;
USR = 1.1.1.1;
USR = 1.1.2;
USR = 1.1.2.1;
USR = 1.1.2.1.6;
USR = 1.5;
USR = 1.5.0;
USR = 1.5.2;
USR = 1.5.2.2;
USR = 1.5.2.2(c);
USR = 1.8;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.4;
USR = 1.8.1.4(c);
USR = 1.8.2;
USR = 1.8.2.11;
USR = 1.8.2.11(a);
USR = 1.8.3;
USR = 1.8.3.1;
USR = 1.8.3.1(a);

Output Flow Dynamics Assumptions:

parking_lot_calculated_occupancy = 1;

1.2.6.1 Maintain Traffic and Sensor Static Data

Input Flows

current_incident_static_data
existing_sensor_static_data
ftop_roadway_characteristics
ftop_static_data
static_data_for_traffic_control_copy

Output Flows

link_data_for_guidance
link_data_update
new_sensor_static_data
request_sensor_static_data
static_data_for_traffic_control_update
static_data_store_updated
supply_incident_static_data

Description:

Overview: This process shall maintain the store of static and link data used by other processes within the Manage Traffic function. Link data shall also be sent to the Provide Driver and Traveler Services function to enable it to obtain data about links that are not in the geographic area which it serves.

Data Flows: All input flows are unsolicited with the exception of 'static_data_for_traffic_control_copy' which is solicited. All output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) communicate with other processes in the Manage Traffic function to obtain their current static data and to provide updates to that data;
(c) when new static data is received and it has been successfully loaded into the store, output the static_data_store_updated data flow so that other processes can receive new copies of the data.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(a);
USR = 1.8.2;
USR = 1.8.2.13;
USR = 1.8.2.13(a);
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(a);

Output Flow Dynamics Assumptions:

link_data_for_guidance = supply_traffic_static_data;
link_data_update = 1/(60*60*24);
new_sensor_static_data = supply_traffic_static_data;
request_sensor_static_data = 6/(60*60*24*7*52);
static_data_for_traffic_control_update = supply_traffic_static_data;
static_data_store_updated = supply_traffic_static_data;
supply_incident_static_data = current_incident_static_data;

1.2.6.2 Provide Static Data Store Output Interface

Input Flows

static_data_for_traffic_control_output
static_data_store_updated

Output Flows

static_data_for_archive
static_data_for_highway_control
static_data_for_highways
static_data_for_parking_lots
static_data_for_ramps
static_data_for_road_control
static_data_for_roads
static_data_for_strategy
static_data_for_vehicle_signage
tmup_map_static_data

Description:

Overview: This process shall provide updates of static data to other processes in the Provide Traffic Control facility of the Manage Traffic function. An update of the data shall only be provided when this process has been notified by another process that the contents of the store of static data has been changed. This process shall provide updates to the map update provider about changes to the static data of a particular region.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'static_data_store_updated'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'static_data_for_highways';
(b) 'static_data_for_highway_control';
(c) 'static_data_for_road_control';
(d) 'static_data_for_parking_lots';
(e) 'static_data_for_ramps';
(f) 'static_data_for_roads';
(g) 'static_data_for_strategy';
(h) 'static_data_for_vehicle_signage';
(i) 'tmup-map_static_data';
(j) 'static_data_for_archive'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flow 'static_data_store_updated';
(b) when the flow in (a) has been received, read the data from the store of static data and send the solicited output flows listed above;

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(a);

Output Flow Dynamics Assumptions:

static_data_for_highways = static_data_store_updated;
static_data_for_highway_control = static_data_store_updated;
static_data_for_road_control = static_data_store_updated;
static_data_for_parking_lots = static_data_store_updated;
static_data_for_ramps = static_data_store_updated;
static_data_for_roads = static_data_store_updated;
static_data_for_strategy = static_data_store_updated;
static_data_for_vehicle_signage = static_data_store_updated;
static_data_for_archive = static_data_store_updated;
tmup-map_static_data = static_data_store_updated;

1.2.7.1 Process Indicator Output Data for Roads

Input Flows

dms_data_for_roads
fmmc_crossing_status_for_roads
har_data_for_roads
hri_device_control
indicator_control_data_for_roads
indicator_preemption_override_for_roads
local_sensor_data_for_roads
train_sense_data

Output Flows

dms_status_for_roads
har_fault_data_for_roads
har_status_for_roads
hri_device_sense
indicator_input_data_from_roads
indicator_response_data_for_roads
intersection_state_data
tbv_har_broadcast_for_roads
td_dms_indication_for_roads
td_lane_use_indication_for_roads
td_signal_indication
tmmc_crossing_clear_at_roads
tmmc_road_equipment_status
tmmc_stop_alternate_mode_at_roads
tp_cross_request_received
tp_cross_road
tp_dms_indication

Description:

Overview: This process shall implement the indicator output data generated by other processes within the Manage Traffic function for use on the roads (surface streets) served by the function. It shall perform the functions needed to provide control at intersections or pedestrian crossings, generate the output for dynamic message signs (dms) and highway advisory radios (HAR), or provide the interface for data to be sent to the units (or systems) that manage multimodal crossings. The dms may be either those that display variable text messages, or those that have fixed format display(s)(e.g. vehicle restrictions, or lane open/close).

Data Flows: All input data flows are unsolicited inputs and all output data flows are solicited outputs.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) provide output data in a form, which is easily understood by drivers and/or travelers, appears in a safe sequence, is unambiguous and does not provide conflicting instructions to drivers and travelers that are likely to result in circumstances which are life threatening;
(c) all output must be maintained for a time period which is sufficient to enable them to be read, understood and reacted to, but not so long that they cause any new indication to be ignored;
(d) if no input of control data is received for a continuous period of time to be locally determined, the process shall start to change its outputs based on local sensor data, and shall clear (blank) the outputs containing advisory message texts;
(e) the outputs to the multimodal crossings shall be maintained for as long as the appropriate control signal is received from other processes, and if no such signals are being received shall be set to null, i.e. the multimodal crossing equipment is not expected to take any action.

User Service Requirements:

USR = 1.0;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(a);
USR = 1.6;
USR = 1.6.0;
USR = 1.6.3;
USR = 1.6.3.3;
USR = 1.6.3.3.1;
USR = 1.6.3.3.2;

USR = 1.6.3.3.4;
USR = 1.6.3.4;
USR = 1.6.3.4(a);
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.1;
USR = 1.10.3.3;
USR = 1.10.3.3.2;
USR = 1.10.3.3.3;
USR = 1.10.4;
USR = 1.10.4.1;
USR = 1.10.5;
USR = 1.10.5.1;
USR = 1.10.5.2;
USR = 1.10.5.2.2;

Output Flow Dynamics Assumptions:

indicator_input_data_from_roads = 1;
indicator_response_data_for_roads = 1;
intersection_state_data = 1;
td-lane_use_indication_for_roads = 1;
td-signal_indication = 1;
td-dms_indication_for_roads = 1;
tmmc-crossing_clear_at_roads = 1;
tmmc-stop_alternate_mode_at_roads = 1;
tmmc-road_equipment_status = 1;
tp-cross_request_received = 1;
tp-cross_road = 1;
tp-dms_indication = 1;
hri_device_sense = 1;
har_status_for_roads = 1;
har_fault_data_for_roads = 1;
tv-har_broadcast_for_roads= har_data_for_roads;
dms_status_for_roads = 1;
tbv-har_broadcast_for_roads = 1;

1.2.7.2 Monitor Roadside Equipment Operation for Faults

Input Flows

har_fault_data_for_highways
har_fault_data_for_roads
indicator_control_monitoring_data_for_highways
indicator_control_monitoring_data_for_roads
indicator_monitoring_suspend
indicator_response_data_for_highways
indicator_response_data_for_roads
vehicle_sign_data_output_fault
vehicle_smart_probe_data_output_fault

Output Flows

information_device_fault_status
traffic_control_device_status

Description:

Overview: This process shall monitor the operation of the processes that output in-vehicle signage, highway advisory radio, as well as indicator data in the road (surface street) and freeway network. It shall report any instances where the indicator response does not match that expected from the contents of the indicator control data it is receiving, the in-vehicle signage process reports a fault, or the HAR processes report a fault.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'indicator_control_monitoring_data_for_highways';
(b) 'indicator_control_monitoring_data_for_roads';
(c) 'indicator_monitoring_suspend';
(d) 'indicator_response_data';
(e) 'vehicle_sign_data_output_fault';
(f) 'vehicle_smart_probe_data_output_fault'
(g) 'har_fault_data_for_roads'
(h) 'har_fault_data_for_highways'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'traffic_control_device_status'
(b) information_device_fault_status'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) output the 'traffic_control_device_status' data flow if the indicator response data does not match the control monitoring data for a locally determined period of time, and a locally determined period of time has elapsed since the control monitoring data last changed;
(c) report all in-vehicle signage and probe data processing faults as soon as they are detected.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

traffic_control_device_status = 1/YEAR;
information_device_fault_status = 1/60;

1.2.7.3 Manage Indicator Preemptions

Input Flows

emergency_vehicle_preemptions
transit_vehicle_roadway_preemptions

Output Flows

indicator_monitoring_suspend
indicator_preemption_override_for_highways
indicator_preemption_override_for_roads

Description:

Overview: This process shall receive indicator (e.g. signal) preemption and priority requests from other functions within ITS. These requests shall enable the process to give selected vehicles (e.g. those that belong to Transit Authorities or Emergency Services) signal preemption or priority at intersections, pedestrian crossings and multimodal crossings in the surface street and freeway network served by the instance of the Manage Traffic function. Sending of the priority request output shall also generate an output to the monitoring process to suspend its activities while the priority request is being served. This process shall only generate its data flow outputs when input data is received.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'emergency_vehicle_preemptions';
(b) 'transit_vehicle_roadway_preemptions'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_monitoring_suspend';
(b) 'indicator_preemption_override_for_highways';
(c) 'indicator_preemption_override_for_roads'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) maintain both output flows for as long as any of the input flows are present;
(c) remove the output flows when the input flows cease to exist.

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.2;
USR = 1.8.2.8;
USR = 1.8.2.8(a);
USR = 1.8.2.8(b);
USR = 1.8.2.8(c);
USR = 1.8.3;
USR = 1.8.3.1;
USR = 1.8.3.1(c);
USR = 5.0;
USR = 5.2;
USR = 5.2.3;

Output Flow Dynamics Assumptions:

indicator_monitoring_suspend = 1;
indicator_preemption_override_for_highways = 1;
indicator_preemption_override_for_roads = 1;

1.2.7.4 Process In-vehicle Signage Data

Input Flows

vehicle_sign_data

Output Flows

vehicle_sign_data_output_fault
vehicle_signage_data

Description:

Overview: This process shall output data for use by in-vehicle signage equipment on vehicles traveling along the road (surface street) and freeway network served by the Manage Traffic function. This data shall be able to provide information from any of the types of indicators that are supported by the function, e.g. intersection controller, pedestrian controller, dynamic message sign (dms), plus data about incidents and link information such as speed, travel times or roadway conditions. The process shall be responsible for its own fault monitoring, which shall check that output data is being sent and that it is an accurate representation of the input data. When a fault is detected this process shall report it to the process responsible for the monitoring of roadside equipment faults.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'vehicle_sign_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'vehicle_signage_data';
(b) 'vehicle_sign_data_output_fault'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) continuously generate the 'vehicle_signage_data' solicited output flow, making sure that it accurately reflects the data received in (a);
(c) detect any processing faults, such as input data does not match output data, or data output failed;
(d) if a fault is detected in (c), send the 'vehicle_sign_data_output_fault' solicited output flow to the fault monitoring process.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

vehicle_signage_data = vehicle_sign_data;
vehicle_sign_data_output_fault = 1/(60*60*24*7);

1.2.7.5 Process Indicator Output Data for Freeways

Input Flows

dms_data_for_highways
fmmc_crossing_status_for_highways
har_data_for_highways
indicator_control_data_for_highways
indicator_preemption_override_for_highways
local_sensor_data_for_highways

Output Flows

dms_status_for_highways
har_fault_data_for_highways
har_status_for_highways
indicator_input_data_from_highways
indicator_response_data_for_highways
tbv_har_broadcast_for_highways
td_dms_indication_for_highways
td_lane_use_indication_for_highways
td_ramp_state_indication
tmmc_crossing_clear_at_highways
tmmc_highway_equipment_status
tmmc_stop_alternate_mode_at_highways

Description:

Overview: This process shall implement the indicator output data generated by other processes within the Manage Traffic function for use on freeways served by the function. It shall perform the functions needed to output control data to ramp metering controllers and multimodal crossings, generate the output for dynamic message signs (dms), or generate the output for highway advisory radios(HAR). The dms may be either those that display variable text messages, or those that have fixed format display(s), for such things as vehicle restrictions, or lane open/close.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'indicator_control_data_for_highways';
(b) 'indicator_preemption_override_for_highways';
(c) 'local_sensor_data_for_highways';
(d) 'har_data_for_highways';
(e) 'fmmc-crossing_status_for_highways'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_input_data_from_highways';
(b) 'indicator_response_data_for_highways';
(c) 'td-lane_use_indication_for_highways';
(d) 'td-ramp_state_indication';
(e) 'td-dms_indication_for_highways';
(f) 'tmmc-crossing_clear_at_highways';
(g) 'tmmc-stop_alternate_mode_at_highways';
(h) 'tbv-har_broadcast_for_highways'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) provide output data in a form which is easily understood by drivers and/or travelers, appears in a safe sequence, is unambiguous and does not provide conflicting instructions to drivers and travelers that are likely to result in circumstances which are life threatening;
(c) all output must be maintained for a time period which is sufficient to enable them to be read, understood and reacted to, but not so long that they cause any new indication to be ignored;
(d) if no input of fresh control data is received for a locally determined continuous period of time, the process shall start to change its outputs based on local sensor data, and shall clear (blank) the outputs containing advisory message texts.

User Service Requirements:

USR = 1.0;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(a);
USR = 1.6;

USR = 1.6.0;
USR = 1.6.3;
USR = 1.6.3.3;
USR = 1.6.3.3.2;
USR = 1.6.3.3.3;
USR = 1.6.3.4;
USR = 1.6.3.4(b);
USR = 1.6.3.4(c);

Output Flow Dynamics Assumptions:

indicator_input_data_from_highways = 1;
indicator_response_data_for_highways = 1;
har_status_for_highways = 1;
har_fault_data_for_highways = 1;
dms_data_for_highways = 1;
td-lane_use_indication_for_highways = 1;
td-ramp-state_indication = 1;
td-dms_indication_for_highways = 1;
tmmc-crossing_clear_at_highways = 1;
tmmc-stop_alternate_mode_at_highways = 1;
tmmc-highway_equipment_status = 1;
tbv-har-broadcast_for_highways = har_data_for_highways;
dms_status_for_highways = 1;

1.2.7.6 Provide Intersection Collision Avoidance Data

Input Flows

intersection_state_data
local_sensor_data_for_roads

Output Flows

intersection_collision_avoidance_data

Description:

Overview: This process shall provide collision avoidance data to vehicles that are approaching intersections served by the Manage Traffic function. The process shall use the data available from traffic sensors to determine any vehicle position conflict(s) that will arise if no action is taken. This process shall output data giving the direction from which the potential collision hazard will arise to the vehicle(s) that is(are) likely to receive the impact.

(a) 'intersection_state_data';
(b) 'local_sensor_data_for_roads'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'intersection_collision_avoidance_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) use the 'intersection_state_data' input flow to develop a model of the intersection controller operation;
(c) process the 'local_sensor_data_for_roads' input flow to determine if there are any vehicles for which a collision is likely, in the context of the current and expected state of the intersection;
(d) if any vehicles are found to have a collision threat, continuously generate the solicited output flow 'intersection_collision_avoidance_data' and send it to the vehicle(s) that is(are) involved;
(e) discontinue the output 'intersection_collision_avoidance_data' when the threat of collision to the vehicle(s) has ceased.

User Service Requirements:

USR = 5.0;
USR = 5.2;
USR = 5.2.3;
USR = 1.10.3.2;

Output Flow Dynamics Assumptions:

intersection_collision_avoidance_data = 1;

1.2.7.7 Process Vehicle Smart Probe Data for Output

Input Flows

vehicle_smart_probe_data_indication

Output Flows

vehicle_smart_probe_data_output
vehicle_smart_probe_data_output_fault

Description:

Overview: This process shall output data about the conditions on roads and freeways. The process shall be provided with this data by other processes in the Manage Traffic function, which have received and processed data output by smart probes in vehicles. The data shall be output by the process for reception by those vehicles that are passing the deployed instance of this process (e.g. by dedicated short range communications). The process shall perform its own fault detection and report faults that are found to the fault monitoring process.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'vehicle_smart_probe_data_indication'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'vehicle_smart_probe_data_output';
(b) 'vehicle_smart_probe_data_output_fault'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) continuously generate the 'vehicle_smart_probe_data_output' solicited output flow, making sure that it accurately reflects the data received in (a);
(c) detect all processing faults, such as input data does not match output data, data output failed, input data not received, input data not valid;
(d) as soon as a fault is detected in (c), generate the 'vehicle_smart_probe_data_output_fault' solicited output flow.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.2;
USR = 1.6.2.2;
USR = 1.6.2.3;
USR = 1.6.2.3.1;
USR = 1.6.2.3.2;
USR = 1.6.2.4;
USR = 1.6.2.4.1;
USR = 1.6.2.5;

Output Flow Dynamics Assumptions:

vehicle_smart_probe_data_output = vehicle_smart_probe_data_indication;
vehicle_smart_probe_data_output_fault = 1/YEAR;

1.2.8.1 Collect Indicator Fault Data

Input Flows

indicator_data_fault_for_highways
indicator_data_fault_for_roads
information_device_fault_status
traffic_control_device_status

Output Flows

indicator_new_fault_update

Description:

Overview: This process shall collect data about faults in the operation of indicators (e.g. signals, dms, har) that have been detected by processes in other parts of the Manage Traffic function. It shall be possible for the faults to be detected locally at the indicators, or centrally through communications links with the indicators.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'indicator_data_fault_for_highways';
(b) 'indicator_data_fault_for_roads';
(c) 'traffic_control_device_status'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_new_fault_update'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the data flows in (a) are received, pass the data on to another process for storage and the activation of fault reporting processes.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

indicator_new_fault_update=(INTERSECTIONS+PEDESTRIANS+CROSSINGS+SIGNS+RAMPS)*1/(60*60*24*7*52)

1.2.8.2 Maintain Indicator Fault Data Store

Input Flows

indicator_current_fault_update
indicator_current_faults_list
indicator_fault_clearance_update
indicator_new_fault_data
indicator_new_fault_update

Output Flows

indicator_current_fault_data
indicator_current_faults_list
indicator_fault_state
indicator_new_fault

Description:

Overview: This process shall collect data about indicator faults that have been detected by processes in other parts of the Manage Traffic function. It shall be possible for the faults to have been detected locally at the indicators, or centrally through communications links with the indicators. The process shall pass on new fault data to another process for communication to the Construction and Maintenance terminator and shall receive fault clearances from the same process communicating with that terminator. It shall also maintain a store of the current fault state of all indicators. The process shall provide facilities that enable traffic operations personnel to review and update the current fault status of all indicators. Details of faulty and fixed equipment shall be passed by the process to the traffic control strategy selection process so that it can adjust its strategy to take account of the current fault(s).

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'indicator_current_fault_update';
(b) 'indicator_new_fault_update';
(c) 'indicator_fault_clearance_update';
(d) 'indicator_new_fault_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_current_fault_data';
(b) 'indicator_current_faults_list';
(c) 'indicator_fault_state';
(d) 'indicator_new_fault'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) the process shall be responsible for the maintenance of the store of the current indicator fault state data, modifying the store as necessary based on the unsolicited input data flows;
(c) If the indicator_current_faults_list data store changes, update and issue the solicited output flows.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

indicator_current_fault_data = 5/(60*60);
indicator_current_faults_list =
(INTERSECTIONS+PEDESTRIANS+CROSSINGS+SIGNS+RAMPS)*1/(60*60*24*7*52);
indicator_fault_state =
(INTERSECTIONS+PEDESTRIANS+CROSSINGS+SIGNS+RAMPS)*2/(60*60*24*7*52);

1.2.8.3 Provide Indicator Fault Interface for C and M

Input Flows

fcm_fault_clearance
indicator_new_fault

Output Flows

indicator_fault_clearance_update
tcm_fault_data

Description:

Overview: This process shall provide an interface for the exchange of data with the Construction and Maintenance terminator. The interface shall be used to both send data containing details of new indicator equipment faults, and to receive clearances when the faults are cleared. The details of new equipment faults and the clearances shall be received from and sent to another process.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'indicator_new_fault'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the construction and maintenance terminator:
(a) 'fcm-fault_clearance'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_fault_clearance_update';
(b) 'tcm-fault_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) on receipt of 'indicator_new_fault', forward the fault in the data flow 'tcm-fault_data' to the appropriate instance of the Construction and Maintenance terminator.
(c) on receipt of the 'fcm-fault_clearance' data flow from the Construction and Maintenance terminator, issue the data flow 'indicator_fault_clearance_update'.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

indicator_fault_clearance_update =
(INTERSECTIONS+PEDESTRIANS+CROSSINGS+SIGNS+RAMPS)*1/(60*60*24*7*52);
tcm-fault_data = (INTERSECTIONS+PEDESTRIANS+CROSSINGS+SIGNS+RAMPS)*1/(60*60*24*7*52);

1.2.8.4 Provide Traffic Operations Personnel Indicator Fault Interface

Input Flows

ftop_indicator_fault_data_input
ftop_indicator_fault_data_request
ftop_indicator_fault_data_update
indicator_current_fault_data

Output Flows

indicator_current_fault_update
indicator_new_fault_data
ttop_current_indicator_faults

Description:

Overview: This process shall provide the interface through which traffic operations personnel access data about faults on indicator equipment controlled by the Manage Traffic function. The process shall enable the personnel to monitor all indicator equipment faults that have been detected, and if necessary, amend that data. It shall also enable the traffic operations personnel to manually input faults in cases where they cannot otherwise be detected.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ftop-indicator_faut_data_input';
(b) 'ftop-indicator_fault_data_request';
(c) 'ftop-indicator_fault_data_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'indicator_current_fault_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'indicator_current_fault_update';
(b) 'indicator_new_fault_data';
(c) 'ttop-current_indicator_faults'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) be capable of accepting input from Traffic Operations Personnel;
(c) be capable of carrying out its own verification of input data received from Traffic Operations Personnel and generating the correct solicited output data flow as a result of input data being received;
(d) as part of the output generation process, carrying out checks for data out of range, missing or spurious values and requesting re-input where required;
(e) providing all output to Traffic Operations Personnel in a form that is readily understood by a human operator.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

indicator_current_fault_update = 1/(60*60*24*7);
indicator_new_fault_data = 1/(60860*24*7*52);
ttop-current_indicator_faults = 5/(60*60);

1.3.1.1 Analyze Traffic Data for Incidents

Input Flows

current_road_network_use
hri_incident_data
incident_analysis_data
static_data_for_incident_management
traffic_image_data
unusual_data

Output Flows

possible_detected_incidents
reversible_lane_status

Description:

Overview: This process shall analyze traffic sensor data, vehicle probe data, or video images for anomalies that could indicate occurrence of an incident. The data may be collected from roads(surface street) and/or highways served by the Manage Traffic function. The process shall pass on any anomalies that it detects to another process in the Manage Incidents facility as possible detected incidents.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'incident_analysis_data';
(b) 'current_road_network_use';
(c) 'traffic_image_data';
(d) 'unusual_data'.

Solicited Input Processing: This process shall receive the following data flows from a local data store:
(a)'static_data_for_incident_management'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above unsolicited inputs being received:
(a)'possible_detected_incidents';
(b)'reversible_lane_status'.

Functional Requirements: This process shall:
(a) run whenever any of the unsolicited data flows listed above is received;
(b) analyze the unsolicited data and identify any anomalies and their location which indicate that traffic is not flowing as expected;
(c) when anomalies in the traffic flow are detected in (b), report them as possible incidents using the solicited output data flow 'possible_detected_incidents' and 'reversible_lane_status'.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.2;
USR = 1.7.1.2.1;
USR = 1.7.1.2.1(e);

Output Flow Dynamics Assumptions:

possible_detected_incidents = 5/(60*60);
reversible_lane_status = 5/(60*60);

1.3.1.2 Maintain Static Data for Incident Management

Input Flows

static_data_for_incident_management
supply_incident_static_data

Output Flows

current_incident_static_data
static_data_for_incident_management

Description:

Overview: This process shall maintain the store of static data (data about the location and features of the road or highway links in the transportation network). This data store is used by another process within the Manage Incidents facility to identify and locate incidents. The static data shall be input to this process from another process within the Planning for Deployment function, and it shall be possible for that process to request a copy of the current static data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'supply_incident_static_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'static_data_for_incident_management';
(b) 'current_incident_static_data'.

Functional Requirements: This process shall:
(a) run when either the unsolicited data flow is received;
(b) as updates are made to the incident static data, load the contents into the output data flow current_incident_static_data;
(c) when the 'supply_incident_static_data' unsolicited data flow is received, load the contents into the store of static data, overwriting any data already present;

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

current_incident_static_data = static_data_for_incident_management;
static_data_for_incident_management = supply_incident_static_data;

1.3.1.3 Process Traffic Images

Input Flows

ft_traffic_images
incident_video_image_control

Output Flows

incident_video_image
reversible_lane_video_images
traffic_image_data

Description:

Overview: This process shall process raw traffic image data received from sensors located on the road (surface street) and freeway network served by the Manage Traffic function. The process shall transform the raw data into images that can be sent to another process within the Manage Incidents facility. It shall also act as the control interface through which the images of traffic conditions which are analyzed for incidents can be changed by the traffic operations personnel, who shall also be supplied with images for viewing.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'ft-traffic_images'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'traffic_image_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flow'ft-traffic_images';
(b) transform the data in (a) into a form in which it can be sent for analysis by another process;
(c) send the data generated in (b) to the data analysis process using the solicited output flow 'traffic_image_data';
(d) at the same time as the data in (c) is output, generate the incident image data flow and send it to the traffic operations personnel interface process;
(e) when the video camera control data flow is received, implement the data it contains to effect the required changes to the system operational parameters.

User Service Requirements:

USR = 1.7;
USR = 1.7.1;
USR = 1.7.1.2;
USR = 1.7.1.2.2;
USR = 1.7.1.2.2(a);
USR = 2.0;
USR = 2.2;
USR = 2.2.1;
USR = 2.2.1.2;
USR = 2.2.1.2.1;
USR = 2.2.1.2.1.3;
USR = 2.2.1.2.2;
USR = 2.2.1.2.2.1;
USR = 2.4;
USR = 2.4.2;
USR = 2.4.2.2;

Output Flow Dynamics Assumptions:

traffic_image_data = 60;
incident_video_image = 1;
reversible_lane_video_images = 1/60;

1.3.2.1 Store Possible Incident Data

Input Flows

fcm_incident_information
fep_event_information
fws_current_weather
fws_predicted_weather
logged_special_vehicle_route
media_incident_data_updates
pollution_incident
possible_detected_incidents

Output Flows

possible_incident_data_update
possible_incidents

Description:

Overview: This process shall receive data on possible incidents from other processes within the Manage Incidents facility and other ITS functions. The process shall load all data that it receives into the store of possible incidents. As part of the loading activity, the process shall enter the data into the relevant parts of the standard format for incident data, and shall assign a level of confidence (e.g. related to the source of the data or time of its detection) to that data.

(a) 'logged_special_vehicle_route';
(b) 'fcm-incident_information';
(c) 'fep-event_information';
(d) 'fws-current_weather';
(e) 'fws-predicted_weather';
(f) 'pollution_incident';
(g) 'possible_detected_incidents';
(h) 'media_incident_data_updates'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'possible_incident_data_update';
(b) 'possible_incidents'.

Functional Requirements: This process shall:
(a) run whenever any of the unsolicited data inputs listed above is received;
(b) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of possible_incidents data;
(c) when possible_incident data is being stored, a level of confidence must be attached to it so that the accuracy of the data can be rated according to its source;

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.2;
USR = 1.7.1.1.2(a);
USR = 1.7.1.1.2(b);
USR = 1.7.1.1.2(c);
USR = 1.7.1.1.2(d);
USR = 1.7.1.1.2(e);
USR = 1.7.1.1.3;
USR = 1.7.1.2;
USR = 1.7.1.2.1;
USR = 1.7.1.2.1(e);
USR = 1.7.1.2.1(g);
USR = 1.7.1.2.2;
USR = 1.7.1.2.3;
USR = 1.7.2;
USR = 1.7.2.1;

Output Flow Dynamics Assumptions:

possible_incident_data_update = 12/(60*60);
possible_incidents = 12/(60*60);

1.3.2.2 Review and Classify Possible Incidents

Input Flows

incident_details
operations_incident_data_updates
possible_incident_data_update
possible_incidents
request_possible_incidents_data

Output Flows

current_incidents_new_data
incident_data_update
incident_details_request
planned_event_data
planned_events
planned_events_new_data
possible_incidents_data_output
tcm_incident_confirmation
tcm_request_incident_change
tep_event_confirmation

Description:

Overview: This process shall review input data about possible incidents and provide verification of the incident. The process shall have the capability of using algorithms to automatically identify and verify an incident. The process shall have the capability to classify an incident as current incident or a planned event and shall be load the data into the store of possible incidents as either current incidents or planned events. The process shall report any incidents that it is unable to verify or classify to the traffic operations personnel for manual verification and classification. The process shall allow the traffic operations personnel to request all possible incidents and carry out the verification and classification process manually.

Data Flows: The following input flows are unsolicited: request_possible_incidents_data, incident_details, possible_incidents_data_update. The following input flows are solicited: operations_incident_data_updates, incident_details, and possible_incidents. All outputs are solicited with the exception of incident_details_request which is generated if no input is received in the incident_details for a locally determined period.

Functional Requirements: This process shall:
(a) run when any of the unsolicited data flows described above is received;
(b) be capable of automatically determining which possible incidents can be converted into real incidents (i.e. are not false alarms) and further classifying the real incidents as planned events or current incidents;
(c) the incident classification process shall use the level of confidence data attached to each set of possible incident data;
(d) if the classification cannot be done automatically with a locally determined level of confidence, send the data to the Traffic Operations Personnel via the 'possible_incidents_data_output' output data flow, for manual classification;
(e) where necessary, format the data for a possible incident into the standard form, adding in any missing fields if necessary, and adding in the traffic impact data field;
(f) when a possible incident has been classified: load it into the planned events or current incidents data stores, delete it from the store of possible incidents, send data flows to activate the process responsible for reviewing either planned events or current incidents, and send the appropriate message to other parts of the ITS;
(g) if necessary, update the data retrieved from the store of possible_incidents for a possible incident sent in from the Construction and Maintenance terminator so that it takes place at time(s) that cause the minimum impact to traffic and return the amended data to the Construction and Maintenance terminator via the 'tcm-request_incident_change' output data flow;
(h) new data read from the store of possible_incidents which is found to complement data already in the planned events or current incidents data stores, will be merged, with any additional data items in the new data loaded into the appropriate data store for the incident that is already recorded.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;

USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(a);

Output Flow Dynamics Assumptions:

current_incidents_new_data = 12/(60*60);
incident_details_request = 5/(60*60);
possible_incidents_data_output = 5/(60*60);
planned_events = 12/(60*60);
planned_event_data = 6/(60*60);
incident_data_update = 12/(60*60);
planned_events_new_data = 12/(60*60);
tcm-incident_confirmation = 1/(60*60*24)*INCIDENTS;
tcm-request_incident_change = 1/(60*60*24)*INCIDENTS;
tep-event_confirmation = 1/(60*60);

1.3.2.3 Review and Classify Planned Events

Input Flows

current_incidents_data
current_incidents_request
incident_data_update
incident_response_status
planned_events_data
reclassify_incidents

Output Flows

current_incident_data
current_incidents
current_incidents_data_output
current_incidents_data_request
current_incidents_data_update
planned_event_data_for_vehicle_signage
request_planned_events_data

Description:

Overview: This process shall receive updates of planned events and review the complete list of them to determine when an incident should be reclassified from planned event to current incident. It shall carry out the re-classification process automatically either upon receiving notice that the store of planned events has been updated, or at some periodic rate. The criteria for reclassifying an incident could be that the planned start time of the event has passed. The process shall request details of planned events from the process that manages their data store and shall send details of any new (re-classified) current incidents to the process that manages their data store. It shall also provide updates of planned events and current incidents to other ITS functions, and details of any new planned events to the process responsible for the output of data to vehicle signage functions.

Data Flows: All inputs are unsolicited except for 'planned_events_data' which is solicited as are all outputs.

Functional Requirements: This process shall:
(a) continuously monitor for the unsolicited input data flows listed above;
(b) carrying out the incident re-classification process on receipt of either the 'reclassify_incidents', 'incident_data_update' or 'incident_response_status' data flows, or when planned events are expected to become current, or in the absence of any inputs on a regular (locally determined time interval) basis;
(c) when the 'incident_data_update' unsolicited input data flow indicates that a new planned event has been found, send the incident details to the process that outputs data to roadside signage processes, using the 'planned_event_data_for_vehicle_signage' solicited output data flow;
(d) when the 'incident_data_update' unsolicited input data flow indicates that a new current incident has been found, request the current incidents data and output that for the new incident to the process responsible for providing incident responses;
(e) automatically re-classify incidents from planned events to current incidents based on the time at which the incident is expected to take place;
(f) when an incident is re-classified from planned event to current incident, send out the data for the new current incident to other parts of ITS, and the data flow to activate the process responsible for responding to incidents.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.2;
USR = 1.7.1.2.2;
USR = 1.7.1.2.2(a);
USR = 1.7.1.2.2(b);
USR = 1.7.4;

Output Flow Dynamics Assumptions:

current_incident_data = 12/(60*60);
current_incidents_data_output = 12/(60*60);
current_incidents = 20/(60*60)*TRAFFIC_OPS + 20/(60*60)*MEDIA_OPS;
current_incidents_data_update = 12/(60*60);
current_incidents_data_request = current_incidents_request+incident_response_status;
planned_event_data_for_vehicle_signage = incident_data_update;
request_planned_events_data = incident_data_update;

1.3.2.4 Provide Planned Events Store Interface

Input Flows

other_planned_events
planned_events_new_data
planned_events_store
request_local_planned_events_data
request_planned_events_data

Output Flows

planned_events_data
planned_events_data_output
planned_events_local_data
planned_events_store
request_other_planned_events_data

Description:

Overview: This process shall provide the interface to, and manage the use of the store containing details of planned events. The process shall enter details of all new planned events into the store, retrieve details on request, and delete details of an incident when it has been re-classified as a current incident.. The process shall be able to receive details of planned events from within the local Manage Incidents facility, and from similar facilities in other Traffic Management Subsystems (TMS's). When requested, the process shall also be able to provide details of its planned events to the Manage Incidents facilities in other TMS's.

Data Flows: All inputs are unsolicited with the exception of 'planned_events_store' and 'other_planned_events' which are solicited. All outputs are solicited with the exception of 'request_other_planned_events_data', an unsolicited output generated regardless of the inputs received.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) the incident data shall be stored and accessed from the 'planned_events_store';
(c) each time data about a new planned event is loaded into the 'planned_events_store', the process shall also pass that data on to other parts of the Manage Traffic function through the output of the planned events data flow;
(d) if the incident may affect traffic outside the local geographic or jurisdictional area served by the instance of the function, then the data about the incident shall be sent to other TMS's using the 'planned_events_data_output' data flow;
(e) when initially run, request data on planned events that may affect local traffic from other TMS's using the 'request_other_planned_events_data' data flow;
(f) when data about planned events in geographic or jurisdictional areas served by other TMS's is received, it shall be loaded into the store of planned events and processed as though the incident(s) had just occurred;
(g) when a request for local planned event data is received, only data on those planned events that may affect traffic outside the geographic or jurisdictional area served by the instance of the function shall be retrieved from the data store and sent to the requesting TMS in the 'planned_events_local_data' data flow.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

planned_events_data = 1/(60*60*24);
planned_events_data_output = 1/(60*60*24);
planned_events_local_data = 1/(60*60*24);
planned_events_data_store = 1/(60*60*24);
request_other_planned_events_data = 1/(60*60*24);

1.3.2.5 Provide Current Incidents Store Interface

Input Flows

current_incidents_data_request
current_incidents_data_update
current_incidents_new_data
current_incidents_store
other_current_incidents
request_local_current_incidents_data

Output Flows

current_incidents_data
current_incidents_store
request_other_current_incidents_data

Description:

Overview: This process shall provide the interface to, and manage the use of the store of current incident details. The process shall enter the details of all new current incidents into the store, retrieve details on request, and delete details of incidents when they cease to be current. The process shall be able to receive details of current incidents from within the local Manage Incidents facility, and from similar facilities in other Traffic Management Subsystems (TMS's). When requested, the process shall also be able to provide details of its current incidents to the Manage Incidents facilities in other TMS's.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'current_incidents_data_update';
(b) 'current_incidents_data_request';
(c) 'current_incidents_new_data';
(d) 'arequest_local_current_incidents_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'current_incidents_store' - which contains data retrieved from a data store;
(b) 'other_current_incidents' - which contains data received from another process.

Unsolicited Output Processing: This process shall provide the following output flow regardless of any input flows that are received:
(a) 'request_other_current_incidents_data';

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'current_incidents_data';
(b) 'current_incidents_store';
(c) 'request_other_current_incidents_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) each time data about a new current incident is received via either the 'current_incidents_data_update' or 'current_incidents_new_data' unsolicited input data flows, it shall be loaded into the store;
(c) when initially run, request data on current incidents that may affect local traffic from other TMS's using the 'request_other_current_incidents_data' data flow;
(d) when data about current incidents in geographic or jurisdictional areas served by other TMS's is received, it shall be loaded into the store of current incidents and processed as though the incident(s) had just occurred;

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

current_incidents_data = current_incidents_data_request;
current_incidents_store = current_incidents_data_update+current_incidents_new_data;
request_other_current_incidents_data = 1/(60*60*24);

1.3.3 Respond to Current Incidents

Input Flows

current_incidents_data_output
defined_responses_data

Output Flows

current_incident_data_for_vehicle_signage
cv_incident_override
dms_updates_for_highways
dms_updates_for_roads
hri_strategy_override
incident_alert
incident_response_clear
incident_response_log
incident_strategy_override
undefined_incident_response

Description:

Overview: This process shall provide responses to incidents that become current, i.e. active. Three general strategies for response to incidents can be supported by the process: 1) Operator enters response (there is no set of predetermined responses), 2) the operator selects response from a set of predetermined responses (possibly modifying the response), and 3) the process automatically accesses and implements a response from a set of predetermined responses (while informing the operator of the actions taken). Where predetermined responses are utilized, the operator shall have the capability to view, modify, or override The predetermined response. The predetermined response to each type of incident shall be defined for the process in the store defined_responses_data. If the process cannot find a predetermined response for a particular incident, it shall send the details of the incident to the traffic operations personnel so that they can provide an update to the store of predetermined responses. The process shall output the predetermined responses to an incident when it receives notification from another process in the Manage Incidents facility that a new current incident has occurred. At the same time it shall also output the incident data to the process responsible for providing broadcast data to roadside processes. The other process in the Manage Incidents facility shall also provide details of incidents that have ceased to be current (terminated) so that this process can send out data to clear the actions requested and roadside broadcast information output in response to its occurrence.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'current_incidents_data_output'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'defined_responses_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'current_incident_data_for_vehicle_signage';
(b) 'incident_alert';
(c) 'incident_response_clear';
(d) 'incident_response_log';
(e) 'incident_strategy_override';
(f) 'cv_incident_override':
(g) 'undefined_incident_response';
(h) 'dms_updates_for_highways';
(i) 'dms_updates_for_roads'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flow 'current_incidents_data_output';
(b) analyze the current incident data against the data in the store of defined responses to determine the appropriate response;
(c) generate the appropriate solicited output flows listed above as a result of determining the appropriate defined response to an incident;
(d) generate the appropriate clearance data in the solicited output flows listed above when the duration of an incident expires;
(e) if a defined response is not found for any incident, then the process shall send data about the incident to the Provide Operator Interfaces for Incidents facility and take no further action;

User Service Requirements:

USR = 1.0;

USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.2;
USR = 1.7.1.2.3;
USR = 1.7.2;
USR = 1.7.2.2;
USR = 1.7.2.3;
USR = 1.7.2.4;
USR = 1.7.2.5;
USR = 1.7.3;
USR = 1.7.3.1;
USR = 1.7.3.1(a);
USR = 1.7.3.2;
USR = 1.7.3.3;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

current_incident_data_for_vehicle_signage = 12/(60*60);
cv_incident_override = logged_hazmat_route;
incident_alert = 1/(60*60);
incident_response_clear = 1/(60*60);
incident_response_log = 1/(60*60);
incident_strategy_override = 1/(60*60);
undefined_incident_response = 1/(60*60*24);
dms_updates_for_highways = 1/(60*60);
dms_updates_for_roads = 1/(60*60);
hri_strategy_override = 1/(60*60);

1.3.4.1 Retrieve Incident Data

Input Flows

current_incidents
map_data_for_incident_display
planned_events_data_output
possible_incidents_data_output
request_incident_media_data
request_incident_operations_data

Output Flows

current_incidents_request
request_possible_incidents_data
retrieved_incident_media_data
retrieved_incident_operations_data

Description:

Overview: This process shall retrieve incident data from the stores of planned events and current incidents that are managed by other processes in the Manage Incidents facility of the Manage Traffic function. The process shall retrieve data as the result of a request which may come from the traffic operations personnel or the media operator. The output shall be returned to the source of the request, except where the media operator has specified that the data should be output to the media system.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_incident_operations_data';
(b) 'request_incident_media_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'current_incidents';
(b) 'planned_events_data';
(c) 'possible_incidents_data';
(d) 'map_data_for_incident_display'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'current_incidents_request';
(b) 'request_possible_incidents_data';
(c) 'retrieved_incident_media_data';
(d) 'retrieved_incident_operations_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input data flows 'request_incident_operations_data' and 'request_incident_media_data';
(b) when either of the flows in (a) is received, request the required incident data from the appropriate store interface process using solicited output data flows 'current_incidents_request' or 'request_possible_incidents_data';
(c) when the appropriate solicited input data flow is received in response to (b), integrate the stored map_data_for_incident_display with the incident data if necessary;
(d) when (c) is completed, send the data to the process from which the data flow in (a) was received.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.2;
USR = 1.7.1.2.1;
USR = 1.7.1.2.1(c);
USR = 1.7.1.2.1(d);
USR = 1.7.1.2.1(f);
USR = 1.7.1.2.2;
USR = 1.7.1.2.2(d);
USR = 1.7.4;

Output Flow Dynamics Assumptions:

current_incidents_request = 20/(60*60)*TRAFFIC_OPS+20/(60*60)*MEDIA_OPS;
request_possible_incidents_data = ftop-request_possible_incidents_data;
incident_data_output = 12/(60*60);
retrieved_incident_operations_data = 20/(60*60)*TRAFFIC_OPS;
retrieved_incident_media_data = request_incident_media_data;

1.3.4.2 Provide Traffic Operations Personnel Incident Data Interface

Input Flows

defined_incident_response_data
ftop_defined_incident_response_data_request
ftop_defined_incident_response_data_update
ftop_incident_camera_action_request
ftop_incident_data_amendment
ftop_incident_information_requests
ftop_output_possible_defined_reponses
ftop_request_possible_incidents_data
ftop_resource_request
ftop_update_defined_incident_responses
incident_video_image
operator_log_for_incidents_data
possible_defined_responses_output
remote_video_image_control
retrieved_incident_operations_data
traffic_operations_resource_response
undefined_incident_response
video_camera_control_strategy
wrong_way_vehicle_detection

Output Flows

defined_incident_response_data_request
defined_incident_response_update_request
defined_incident_response_updates
incident_video_image_control
operations_incident_data_updates
operator_log_for_incidents_data
possible_defined_responses_output_request
reclassify_incidents
request_incident_map_display_update
request_incident_operations_data
traffic_operations_resource_request
ttop_defined_incident_responses_data
ttop_incident_information_display
ttop_incident_video_image_output
ttop_possible_defined_response_output
ttop_possible_incidents_data
ttop_resource_response
ttop_undefined_response_details
ttop_wrong_way_detection

Description:

Overview: This process shall provide the interface between the traffic operations personnel and the Manage Incidents facility of the Manage Traffic function. It shall enable the personnel to request and amend details of predicted and current incidents and predetermined incident responses, obtain and control incident video image data and manually re-classify incidents as possible or current or a planned event. It shall also output to the traffic operations personnel incident details to which no predetermined response currently exists. The process shall support inputs from and outputs to the traffic operations personnel Where appropriate and/or requested by the traffic operations personnel, the process shall provide the output 'display' in a form incorporating a map of the relevant part(s) of the surface street and freeway network served by the function. The process shall obtain the map from a local data store, which it shall request to be updated by another process as and when required.'retrieved_incident_operations_data', and 'wrong_way_vehicle_detection; which are solicited along with all output flows.

Functional Requirements: This process shall:
(a) continuously monitor the input data flows and provide acknowledgement of receipt of those from Traffic Operations Personnel;
(b) be capable of accepting input from Traffic Operations Personnel;
(c) be capable of carrying out its own verification of input data received from Traffic Operations Personnel and generating the correct solicited output data flow as a result of data being received;
(d) as part of the output generation process, checking for data out of range, missing or spurious values and requesting re-input where required;
(e) provide output to Traffic Operations Personnel in a form that is readily understood by a

human operator;
(f) only generate the outputs listed above as a result of receiving inputs from the Traffic Operations Personnel or other processes;
(g) when the request for changes to the parameters affecting the operation of the sensor systems (e.g. closed circuit television) responsible for providing sensed images of incidents (including but not limited to video) are received from traffic operations personnel ('ftop-incident_camera_action_request'), generate the 'incident_video_image_control' data flow to the image processing facility;
(h) when video æÆ images of incidents are received (incident_video_image), output them as 'ttop-incident_video_image_output' to the traffic operations personnel;
(i) the use of the digitized map display shall be automatic and shall be at a resolution best suited to the quantity and scope of data being displayed, i.e. the map shall be to the largest possible scale.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

defined_incident_response_data_request = 1/(60*60*24);
defined_incident_response_updates = 2/(60*60*24);
defined_incident_response_update_request = 1/(60*60*24*7);
incident_video_image_control = ftop-incident_camera_action_request;
operations_incident_data_updates = 4/(60*60);
possible_defined_responses_output_request = 1/(60*60*24);
reclassify_incidents = 4/(60*60);
request_incident_operations_data = 20/(60*60)*TRAFFIC_OPS;
request_incident_map_display_update = 2/(60*60*24*7*52);
ttop-defined_incident_responses_data = 1/(60*60);
ttop-incident_information_display = 1/(60*60);
ttop-possible_defined_response_output = 1/(60*60*24);
ttop-possible_incidents_data = 1/(60*60);
ttop-undefined_response_details = undefined_incident_response;
ttop-incident_video_image_output = 1;
ttop-resource_response = request_incident_operations_data;
ttop-wrong_way_detection = 1/(60*60);

1.3.4.3 Provide Media Incident Data Interface

Input Flows

fm_incident_data_request
fm_incident_information
retrieved_incident_media_data

Output Flows

media_incident_data_updates
request_incident_media_data
tm_incident_data

Description:

Overview: This process shall provide the interface between the Media and the Manage Incidents facility. It shall enable the media to request details of incidents and shall allow transmission of incident information to the media. The media shall also provide raw input data on possible incidents. The process shall enable the output to incorporate a map of the area to which the incidents relate.

Data Flows: All inputs are unsolicited with the exception of 'retrieved_incident_media_data' which is solicited as are all outputs.

Functional Requirements: This process shall:
(a) continuously monitor the input data flows and acknowledge receipt of those from the Media;
(b) be capable of accepting input from the Media;
(c) be capable of carrying out its own verification of input data received from the Media and generating the correct solicited output data flow as a result of the input data being received;
(d) as part of the output generation process, carry out checks for data out of range, missing or spurious values and request re-input where necessary;
(e) use the 'media_incident_data_updates' solicited output data flow to send data on a possible incident when this possible incident data is received from the Media in 'fm-incident_information';
(f) provide all output to the Media in a form that is readily understood;
(g) only generate the outputs listed above as a result of receiving inputs from the Media or the other processes;
(h) the use of the digitized map display shall be automatic and shall be at a resolution best suited to the quantity and scope of data being displayed, i.e. the map shall be to the largest possible scale.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(d);
USR = 1.7.4;

Output Flow Dynamics Assumptions:

media_incident_data_updates = fm-incident_information;
tm-incident_data = fm-incident_data_request;
request_incident_media_data = fm-incident_data_request;

1.3.4.4 Update Incident Display Map Data

Input Flows

fmup_incident_display_update
request_incident_map_display_update

Output Flows

map_data_for_incident_display
tmup_request_incident_display_update

Description:

Overview: This process shall provide updates to the store of digitized map data used with displays of incident data produced by processes in the Manage Incidents facility of the Manage Traffic function. The process shall obtain the new data from a map provider or other appropriate data source, on receiving an update request from the traffic operations personnel interface process within the Manage Incidents facility. fmup-incident_display_update and all outputs are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for the receipt of the 'request_incident_map_display_update' unsolicited data flow;
(b) when the data flow in (a) is received, generate the 'tmup-request_incident_display_update' solicited output data flow and continuously monitor for receipt of the 'fmup-incident_display_update' solicited input data flow;
(c) when the 'fmup-incident_display_update' flow is received, output the 'map_data_for_incident_display'

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

tmup-request_incident_display_update = 2/(60*60*24*7*52);
map_data_for_incident_display = 2/(60*60*24*7*52);

1.3.4.5 Manage Resources for Incidents

Input Flows

fcm_resource_response
resource_request
traffic_operations_resource_request

Output Flows

operator_log_for_incidents_data
resource_deployment_status
tcm_resource_request
traffic_operations_resource_response

Description:

Overview: This process shall provide the capability for the Manage Traffic function to generate and receive requests for resources in responding to incidents. The process shall provide the capability for traffic operations personnel to request resources from the Construction and Maintenance to provide equipment and support for incident response and clean up. The process shall be able to receive resource requests from the Manage Emergency function and respond with the status of the response by Constructionand Maintenance or the traffic operations personnel.

Data Flows: All input data flows arde unsolicited and all output flows are solicited.;

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input of resource response data is received, generate the output data flows identified above, and create an initial entry in the mangage resource incidents data store;
(c) when other inputs are received, update the data for the resource response to which they relate in the manage resource incidents data store.;

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(c);
USR = 1.7.1.2;
USR = 1.7.1.2.2;
USR = 1.7.1.2.2(e);
USR = 1.7.3;
USR = 1.7.3.1;
USR = 1.7.3.1(b);

Output Flow Dynamics Assumptions:

resource_deployment_status = 1/(60/60);
traffic_operations_resource_response = 1/(60/60);
traffic_operations_resource_request = traffic_operations_resource_response;
tcm-resource_request = resource_deployment_status;

1.3.5 Manage Possible Predetermined Responses Store

Input Flows

defined_incident_response_update_request
possible_defined_responses
possible_defined_responses_data
possible_defined_responses_output_request

Output Flows

defined_incident_response_changes
possible_defined_responses
possible_defined_responses_output

Description:

Overview: This process shall manage the data store containing possible predetermined responses to Incidents used within the Manage Incidents facility. These responses shall be those that another process within the facility has found to be worth including in the store of predetermined responses from an analysis of the incident response log. This process shall enable retrieval of the data from the store for presentation to traffic operations personnel and its possible transfer to the process that manages the store of predetermined incident responses that are actually used by other processes in the Manage Incidents facility.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'defined_incident_response_update_request';
(b) 'possible_defined_responses_data';
(c) 'possible_defined_responses_output_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'possible_defined_responses

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'defined_incident_response_changes';
(b) 'possible_defined_responses_output'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the request is received for the transfer of a possible response to the process managing the store of defined responses ('defined_incident_response_update_request'), the response data shall be deleted from the store of possible responses once the transfer of the 'defined_incident_response_changes'

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

defined_incident_response_changes = defined_incident_response_update_request;
possible_defined_responses = possible_defined_responses_data;
possible_defined_responses_output = 1/(60*60*24);

1.3.6 Manage Predetermined Incident Response Data

Input Flows

defined_incident_response_changes
defined_incident_response_data_request
defined_incident_response_updates
defined_responses_data

Output Flows

defined_incident_response_data
defined_responses_data

Description:

Overview: This process shall manage data held in the store of predetermined incident responses that are used by processes within the Manage Incidents facility of the Manage Traffic function. The process shall provide details of the current predetermined responses in response to requests from traffic operations personnel, and shall also update the store with new responses received from the process that manages the store of possible predetermined responses.

(a) 'defined_incident_response_data_request';
(b) 'defined_incident_response_changes';
(c) 'defined_incident_response_updates'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'defined_responses_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above unsolicited inputs being received:
(a) 'defined_responses_data';
(b) 'defined_incident_response_data'.

Functional Requirements: This process shall:
(a) run whenever any of the unsolicited data flows shown above is received;
(b) if the data flow in (a) is a 'defined_incident_response_data_request' request for data, retrieve it from the store of defined responses and return it to the requesting process in the 'defined_incident_response_data' solicited output flow listed above;
(c) if the data flow in (a) contains new data for the store of defined responses, load it into the store;

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

defined_incident_response_data = 1/(60*60*24);
defined_response_data = 1/(60*60*24);

1.3.7 Analyze Incident Response Log

Input Flows

incident_response_log

Output Flows

possible_defined_responses_data

Description:

Overview: This process shall analyze the data in the log of incident responses within the Manage Incidents facility of the Manage Traffic functions. The process shall analyze the log so that possible standard predetermined incident responses can be identified from the data in the incident_response_log data store. Any such possible standard predetermined responses that are identified shall be passed by this process to the process that manages the store of possible predetermined responses.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'incident_response_log'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'possible_defined_responses_data'.

Functional Requirements: This process shall:
(a) analyze the data in the log of incident responses data to determine any response patterns that could be used as standards for the responses to particular types of incidents;
(b) send identified possible standard defined responses to the process that enables them to be stored and reviewed by the traffic operations personnel.

User Service Requirements:

USR = 1.0;
USR = 1.7;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

possible_defined_responses_data = 1/(60*60*24);

1.4.1 Provide Traffic Operations Personnel Demand Interface

Input Flows

demand_forecast_data
demand_forecast_result
demand_input_data
demand_management_result
demand_policy_data
ftop_demand_data_request
ftop_demand_data_update_request
ftop_demand_forecast_request
ftop_demand_policy_activation
ftop_demand_policy_information_request
ftop_demand_policy_updates
map_data_for_demand_display

Output Flows

demand_data_update_request
demand_forecast_request
demand_management_activate
demand_policy_data
request_demand_display_update
ttop_demand_data
ttop_demand_forecast_data
ttop_demand_forecast_result
ttop_demand_policy_activation_result
ttop_demand_policy_information

Description:

Overview: This process shall provide the interface between the traffic operations personnel and the processes and data stores used within the Manage Demand facility of the Manage Traffic function. It shall enable the traffic operations personnel to access the data used as input by the demand forecasting process and the results of that process, to request that the input data be updated, set the policies used as input to the Calculate Forecast Demand process, to request that the demand forecasting process runs, and to run the process that implements the results. Where appropriate and/or requested by the traffic operations personnel, the process shall provide the output in a form that includes a map of the relevant part(s) of the road and freeway network served by the Manage Travel Demand function. The process shall obtain the map from a local data store, which it shall request to be updated by another process when required.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ftop-demand_policy_information_request';
(b) 'ftop-demand_policy_updates';
(c) 'ftop-demand_data_update_request';
(d) 'ftop-demand_data_request';
(e) 'ftop-demand_forecast_request';
(f) 'ftop-demand_policy_activation;.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'demand_mamagement_result';
(b) 'demand_input_data';
(c) 'demand_policy_data';
(d) 'demand_forecast_data';
(e) 'demand_forecast_result';
(f) 'map_data_for_demand_display'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ttop-demand_policy_information';
(b) 'ttop-demand_data';
(c) 'ttop-demand_forecast_data';
(d) 'ttop-demand_policy_activation_result';
(e) 'ttop-demand_forecast_result';
(f) 'demand_data_update_request';
(g) 'demand_forecast_request';
(h) 'demand_management_activate';

(i) 'request_demand_display_update'.

Functional Requirements: This process shall:
(a) continuously monitor the input data flows and provide acknowledgement of receipt of those from Traffic Operations Personnel;
(b) be capable of accepting input from traffic operations personnel;
(c) be capable of carrying out its own verification of input data received from traffic operations personnel and generating the correct solicited output data flow as a result of the input data being received;
(d) as part of the output generation process, carrying out checks for data out of range, missing or spurious values and requesting re-input where necessary;
(e) providing all output to traffic operations personnel in a form that is readily understood by a human operator;
(f) only generate the outputs listed above as a result of receiving inputs from the traffic operations personnel or the other processes;
(g) as locally determined generate the 'request_demand_display_update' request for new map data;
(h) the use of the map data shall be automatic and shall be at a resolution best suited
to the quantity and scope of data being displayed, i.e. the map shall be to the largest possible scale.

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.1;

Output Flow Dynamics Assumptions:

demand_data_update_request = ftop-demand_data_update_request;
demand_forecast_request = ftop-demand_forecast_request;
demand_management_activate = ftop-demand_policy_activation;
demand_policy_data = ftop-demand_policy_updates;
request_demand_display_update = 2/(60*60*24*7*52);
ttop-demand_policy_information = ftop-demand_policy_information_request;
ttop-demand_data = ftop-demand_data_request;
ttop-demand_forecast_data = ftop-demand_data_request;
ttop-demand_policy_activation_result = ftop-demand_policy_activation;
ttop-demand_forecast_result = ftop-demand_forecast_request;

1.4.2 Collect Demand Forecast Data

Input Flows

current_other_routes_use
current_transit_routes_use
demand_data_update_request
fws_current_weather
fws_predicted_weather
hri_status_for_traffic_demand
parking_lot_charge_details
parking_lot_charge_direct_details
pollution_state_data
toll_price_details
toll_price_direct_details
traffic_data_for_demand
transit_fare_details
transit_fare_direct_details
transit_running_data_for_demand
transit_services_for_demand
unusual_congestion
weather_service_information_request

Output Flows

demand_input_data
parking_lot_charge_direct_request
parking_lot_charge_request
pollution_state_data_request
toll_price_direct_request
toll_price_request
traffic_data_demand_request
transit_conditions_demand_request
transit_fare_direct_request
transit_fare_request
transit_services_demand_request
weather_service_information

Description:

Overview: This process shall collect data from other ITS functions for use as input to the demand forecasting process within the Manage Demand facility of the Manage Traffic function. The process shall support data retrieval from other functions on request from the traffic operations personnel and through the receipt of unsolicited data from ITS functions. It shall load all the data that it receives in a consistent format into the input store used by the demand forecasting process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'unusual_congestion';
(b) 'prediction_data';
(c) 'current_transit_routes_use';
(d) 'current_road_network_use';
(e) 'fws-current_weather';
(f) 'fws-predicted_weather';
(g) 'demand_data_update_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'parking_lot_charge_details';
(b) 'pollution_state_data';
(c) 'price_data';
(d) 'toll_price_details';
(e) 'transit_fare_details';
(f) 'transit_running_data_for_demand';
(g) 'transit_services_for_demand';
(h) 'traffic_data_for_demand'
(i) 'parking_lot_charge_direct_details'
(j) 'toll_price_direct_data'
(k) 'transit_fare_direct_details'.

Unsolicited Output Processing: This process shall periodically generate the following output flows

to other processes and functions within ITS and the local store of input data:
(a) 'pollution_state_data_request';
(b) 'parking_lot_charge_request';
(c) 'toll_price_request';
(d) 'transit_conditions_demand_request';
(e) 'transit_services_demand_request';
(f) 'traffic_data_demand_request';
(g) 'transit_fare_request'
(h) 'transit_fare_direct_request'
(i) 'toll_price_direct_request_'
(j) 'parking_lot_charge_direct_request'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'demand_input_data'.

Functional Requirements: This process shall:
(a) run at locally determined intervals and generate the unsolicited outputs listed above, unless requested to run by the unsolicited input 'demand_data_update_request';
(b) when running, scan all the unsolicited inputs listed above and collect the data that they are currently providing;
(c) when all inputs have been obtained, produce the solicited output shown above to load the collected data into the 'demand_input_data' store;
(d) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of demand input data;

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.1;
USR = 1.8.1.2;
USR = 1.8.1.2(e);
USR = 1.8.1.2(f);
USR = 1.8.1.3;
USR = 1.8.1.3(e);
USR = 1.8.1.3(f);
USR = 1.8.1.4;
USR = 1.8.1.5;
USR = 1.8.1.5(a);
USR = 1.8.1.5(c);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(a);
USR = 1.8.2.10;
USR = 1.8.2.11;
USR = 1.8.2.12;
USR = 1.8.2.13;
USR = 1.8.2.14;
USR = 1.8.2.2;
USR = 1.8.2.3;
USR = 1.8.2.4;
USR = 1.8.2.4(f);
USR = 1.8.2.5;
USR = 1.8.2.5(a);
USR = 1.8.2.5(b);
USR = 1.8.2.5(c);
USR = 1.8.2.5(d);
USR = 1.8.2.5(e);
USR = 1.8.2.6;
USR = 1.8.2.7;
USR = 1.8.2.7(a);
USR = 1.8.2.7(b);
USR = 1.8.2.7(c);

USR = 1.8.2.8;
USR = 1.8.2.8(a);
USR = 1.8.2.9;
USR = 1.8.2.9(a);
USR = 1.8.2.9(b);
USR = 1.8.2.9(c);
USR = 1.8.3;
USR = 3.0;
USR = 3.1;
USR = 3.1.5;
USR = 3.1.5.1;
USR = 3.1.5.1.1;
USR = 3.1.5.2;
USR = 3.1.5.3;

Output Flow Dynamics Assumptions:

demand_input_data = ftop-demand_data_update_request;
parking_lot_charge_request = ftop-demand_data_update_request;
parking_lot_charge_direct_request = ftop -demand_data_update_request;
pollution_state_data_request = ftop-demand_data_update_request;
price_request = ftop-demand_data_update_request;
toll_price_request = ftop-demand_data_update_request;
toll_price_direct_request = ftop- demand_data_update_request;
traffic_data_demand_request = ftop-demand_data_update_request;
transit_conditions_demand_request = ftop-demand_data_update_request;
transit_fare_request = ftop-demand_data_update_request;
transit_fare_direct_request = ftop-demand_data_update_request;
transit_services_demand_request = ftop-demand_data_update_request;
weather_service_information = weather_service_information_request;

1.4.3 Update Demand Display Map Data

Input Flows

fmup_demand_display_update
request_demand_display_update

Output Flows

map_data_for_demand_display
tmup_request_demand_display_update

Description:

Overview: This process shall provide updates to the store of map data used for displays of forecast traffic and travel demand produced by processes in the Manage Travel Demand facility of the Manage Traffic function. The process shall obtain the new data from a specialist map data supplier or some other appropriate source, on receiving an update request from the traffic operations personnel interface process within the Manage Travel Demand facility.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_demand_display_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to external functions:
(a) 'fmup-demand_display_update'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmup-request_demand_display_update';
(b) 'map_data_for_demand_display'.

Functional Requirements: This process shall:
(a) continuously monitor for the receipt of the 'request_demand_display_update' unsolicited data flow;
(b) when the data flow in (a) is received, generate the 'tmup-request_demand_display_update' solicited output data flow and continuously monitor for receipt of the 'fmup-demand_display_update' solicited input data flow;
(c) when the flow in (b) is received, load the 'map_data_for_demand_display' data store;
(d) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of map data;

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.1;

Output Flow Dynamics Assumptions:

tmup-request_demand_display_update = 2/(60*60*24*7*52);
map_data_for_demand_display = 2/(60*60*24*7*52);

1.4.4 Implement Demand Management Policy

Input Flows

demand_forecast_data
demand_management_activate
parking_lot_charge_change_response
toll_price_changes_response
transit_services_changes_response

Output Flows

ahs_control_data
demand_management_result
demand_overrides
parking_lot_charge_change_request
toll_price_changes_request
transit_services_changes_request

Description:

Overview: This process shall implement the traffic and travel demand forecast data produced by the demand forecasting process in the Manage Travel Demand facility of the Manage Traffic function. The new demand forecast data shall be implemented in such a way that it can influence the demand from travelers for various types of services provided by ITS functions. The process shall when required, request changes to transit services, and/or the charges for tolls, and/or the use of parking lot spaces (as per the locally determined demand policy). It shall communicate the results of its policy implementation to the process that provides the interface to the traffic operations personnel.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'demand_management_activate'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'demand_forecast_data';
(b) 'parking_lot_charge_change_response';
(c) 'toll_price_changes_response';
(d) 'transit_services_changes_response'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ahs_control_data';
(b) 'demand_management_result';
(c) 'demand_overrides';
(d) 'parking_lot_charge_change_request';
(e) 'toll_price_changes_request';
(f) 'transit_services_changes_request'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the 'demand_management_activate' unsolicited data;
(b) when the flow in (a) is received, send data to other Manage Traffic facilities and ITS functions using the solicited output data flows listed above;
(c) be capable of interpreting the contents of the store of demand forecast data in a way that the outputs that are generated in (d) are readily understood by the receiving processes;
(d) provide continuous feedback of the responses to the flows in (b) using the demand management result solicited output data flow.

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.1;
USR = 1.8.1.2;
USR = 1.8.1.2(e);
USR = 1.8.1.2(f);
USR = 1.8.1.3;

USR = 1.8.1.3(e);
USR = 1.8.1.3(f);
USR = 1.8.1.4;
USR = 1.8.1.5;
USR = 1.8.1.5(a);
USR = 1.8.1.5(b);
USR = 1.8.1.6;
USR = 1.8.1.6(d);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.4;
USR = 1.8.2.4(f);
USR = 1.8.2.10;
USR = 1.8.2.11;
USR = 1.8.2.12;
USR = 1.8.2.13;
USR = 1.8.2.14;
USR = 1.8.2.14(a);
USR = 1.8.2.14(b);
USR = 1.8.2.14(c);
USR = 1.8.2.2;
USR = 1.8.2.3;
USR = 1.8.2.3(a);
USR = 1.8.2.3(b);
USR = 1.8.2.3(c);
USR = 1.8.2.3(d);
USR = 1.8.2.4;
USR = 1.8.2.5;
USR = 1.8.2.5(a);
USR = 1.8.2.5(b);
USR = 1.8.2.5(c);
USR = 1.8.2.5(d);
USR = 1.8.2.5(e);
USR = 1.8.2.6;
USR = 1.8.2.7;
USR = 1.8.2.7(a);
USR = 1.8.2.7(b);
USR = 1.8.2.7(c);
USR = 1.8.2.8;
USR = 1.8.2.8(a);
USR = 1.8.2.9;
USR = 1.8.2.9(a);
USR = 1.8.2.9(b);
USR = 1.8.2.9(c);
USR = 1.8.3;
USR = 3.0;
USR = 3.1;
USR = 3.1.5;
USR = 3.1.5.1;
USR = 3.1.5.1.1;
USR = 3.1.5.2;
USR = 3.1.5.3;

Output Flow Dynamics Assumptions:

ahs_control_data = ftop-demand_policy_activation;
demand_management_result = ftop-demand_policy_activation;
demand_overrides = ftop-demand_policy_activation;
parking_lot_charge_change_request = ftop-demand_policy_activation;
toll_price_changes_request = ftop-demand_policy_activation;
transit_services_changes_request = ftop-demand_policy_activation;

1.4.5 Calculate Forecast Demand

Input Flows

demand_forecast_request
demand_input_data
demand_policy_data

Output Flows

demand_forecast_data
demand_forecast_result

Description:

Overview: This process shall provide a forecast of traffic and travel demand in the geographic area served by the Manage Traffic function to which this instance of the Manage Travel Demand facility belongs. The process shall base its forecast on the current and predicted traffic levels traveler demand patterns obtained from an analysis of data obtained from elsewhere within the Manage Traffic function and from other ITS functions as well as locally determined demand policy. The process shall produce a demand forecast that changes the way that services are provided by ITS functions according to locally determined demand policy.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'demand_forecast_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'demand_input_data';
(b) 'demand_policy_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'demand_forecast_data';
(b) 'demand_forecast_result'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the 'demand_forecast_request' unsolicited data flow;
(b) when the data flow in (a) is received, use appropriate algorithms to calculate future traffic and travel demand patterns across locally determined modes of transportation using the stores of demand input data and demand policy data;
(c) provide results of the calculation of the new traffic and travel demand forecast in the demand forecast result data flow;

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.1;
USR = 1.8.1.2;
USR = 1.8.1.3;
USR = 1.8.1.4;
USR = 1.8.1.5;
USR = 1.8.1.5(b);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(f);
USR = 1.8.2.10;
USR = 1.8.2.11;
USR = 1.8.2.12;
USR = 1.8.2.13;
USR = 1.8.2.14;
USR = 1.8.2.2;
USR = 1.8.2.3;

Output Flow Dynamics Assumptions:

demand_forecast_data = ftop-demand_forecast_request;
demand_forecast_result = ftop-demand_forecast_request;

1.5.1 Provide Traffic Operations Personnel Pollution Data Interface

Input Flows

ftop_pollution_data_information_request
ftop_pollution_parameter_updates
map_data_for_pollution_display
pollution_reference_data_output
pollution_state_data_output

Output Flows

pollution_reference_data_request
pollution_reference_data_update
pollution_state_data_output_request
request_pollution_map_display_update
ttop_pollution_data_display

Description:

Overview: This process shall provide the interface between the traffic operations personnel and the processes and data stores used within the Manage Emissions facility of the Manage Traffic function. The process shall enable the personnel to access and update the pollution reference data used by other processes within the facility, and to access the pollution state data provided by those processes. The process shall support inputs from the traffic operations personnel. Where appropriate and/or requested by the traffic operations personnel, the process shall incorporate map data of the relevant part(s) of the surface street and freeway network served by the Manage Traffic function. The process shall obtain the map from a local data store, which it shall request to be updated by another process as and when required.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ftop-pollution_data_information_request';
(b) 'ftop-pollution_parameter_updates'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'map_data_for_pollution_display';
(b) 'pollution_reference_data_output';
(c) 'pollution_state_data_output'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'pollution_reference_data_request';
(b) 'pollution_reference_data_update';
(c) 'pollution_state_data_output_request';
(d) 'ttop-pollution_data_display';
(e) 'request_pollution_map_display_update'.

Functional Requirements: This process shall:
(a) continuously monitor the input data flows and provide acknowledgement of receipt of those from Traffic Operations Personnel;
(b) be capable of accepting input from Traffic Operations Personnel;
(c) be capable of carrying out its own verification of input data received from Traffic Operations personnel and generating the correct solicited output data flow as a result of input data being received;
(d) as part of the output generation process, carrying out checks for data out of range, missing or spurious values and requesting re-input where necessary;
(e) provide all output to Traffic Operations Personnel in a form that is readily understood by a human operator;
(f) only generate the outputs listed above as a result of receiving inputs from the Traffic Operations Personnel or the other processes;
(g) use the map data at a resolution best suited to the quantity and scope of data being displayed, i.e. the map shall be to the largest possible scale.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.2;
USR = 1.8.2.2(a);
USR = 1.8.2.2(c);
USR = 1.9.0;
USR = 1.9.1;
USR = 1.9.1.2;
USR = 1.9.1.2.1;

USR = 1.9.1.2.2;
USR = 1.9.2.2.1;
USR = 1.9.2.2.3;

Output Flow Dynamics Assumptions:

pollution_reference_data_request = ftop-pollution_data_information_request;
pollution_reference_data_update = ftop-pollution_parameter_updates;
pollution_state_data_output_request = ftop-pollution_data_information_request;
ttop-pollution_data_display = ftop-pollution_data_information_request;
request_pollution_map_display_update = 2/(60*60*24*7*52);

1.5.2 Process Pollution Data

Input Flows

fe_area_pollutant_levels
pollution_state_roadside_collection
pollution_state_static_acceptance_criteria

Output Flows

archive_pollution_data
current_traffic_pollution_data
pollution_incident
pollution_state_static_collection
pollution_state_static_log_data
tm_pollution_data
wide_area_pollution_data

Description:

Overview: This process shall process the pollution data being collected from sensors in the geographic area being served by the Manage Traffic function. The process shall integrate data from distributed roadside sensors (provided by another process) with that obtained directly from sensors looking at the general (wide area) environment. The data shall be checked by the process against the pollution levels that have been set up as reference points. If the process finds that the detected levels of pollution exceed the reference levels it shall generate pollution warnings. The process shall send these warnings to other processes in the Manage Traffic function for output to drivers and travelers.

(a) 'fe-area_pollutant_levels'
(b) 'pollution_state_static_acceptance_criteria';
(c) 'pollution_state_roadside_collection'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'pollution_state_static_log_data';
(b) 'pollution_incident';
(c) 'pollution_state_static_collection';
(d) 'wide_area_pollution_data';
(e) 'tm-pollution_data';
(f) 'current_traffic_pollution_data';
(g) 'archive_pollution_data'>.

Functional Requirements: This process shall:
(a) monitor the unsolicited input data flows shown above;
(b) process the 'fe-area_pollutant_levels' unsolicited data flow and analyze the pollutant levels it is providing;
(c) use the data obtained from the 'pollution_state_static_acceptance_criteria' unsolicited input data flow to determine whether or not the levels of the pollutants exceed that which is considered either safe and/or desirable;
(d) continuously output values of the current pollution levels to the process that manages the store of pollution data state via the 'pollution_state_static_collection' solicited output data flow, and to other parts of ITS via the 'wide_area_pollution_data' solicited output data flow;
(e) if the pollution level(s) are found to be unsafe, declare an incident by sending the 'pollution_incident' unsolicited data flow to the Manage Incidents facility, isolating it to individual sectors within the geographic area covered by ITS;
(f) periodically the process shall send the current pollution levels to the log of pollution data using the 'pollution_state_static_log_data' solicited output data flow.

User Service Requirements:

USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.4;
USR = 1.8.1.4(d);
USR = 1.9.0;
USR = 1.9.1;
USR = 1.9.1.1;
USR = 1.9.1.1.1;
USR = 1.9.1.1.2;
USR = 1.9.1.1.3;
USR = 1.9.1.2;

USR = 1.9.1.2.1;
USR = 1.9.1.2.2;

Output Flow Dynamics Assumptions:

pollution_state_static_log_data = 12/(60*60);
pollution_state_static_collection = 1;
pollution_incident = 4/(60*60);
wide_area_pollution_data = 12/(60*60);
tm-pollution_data = 12/(60*60);
current_traffic_pollution_data = 12/(60*60);
archive_pollution_data = 12/(60*60);

1.5.3 Update Pollution Display Map Data

Input Flows

fmup_pollution_display_update
request_pollution_map_display_update

Output Flows

map_data_for_pollution_display
tmup_request_pollution_display_update

Description:

Overview: This process shall provide updates to the map data used in displays of pollution data produced by processes in the Manage Emissions facility of the Manage Traffic function. The process shall obtain the map data from a specialist map data supplier or some other appropriate data source, on receiving an update request from the traffic operations personnel interface process within the Manage Emissions facility.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_pollution_display_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to external functions:
(a) 'fmup-pollution_display_update'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmup-request_pollution_display_update';
(b) 'map_data_for_pollution_display'.

Functional Requirements: This process shall:
(a) continuously monitor for the receipt of the 'request_pollution_display_update' unsolicited data flow;
(b) when the data flow in (a) is received, generate the first solicited output data flow shown above and continuously monitor for receipt of the solicited input data flow shown above;
(c) when the flow in (b) is received, output the 'tmup-request_pollution_display_update' solicited output data flow shown above;
(d) be capable of receiving the 'fmup-pollution_display_update' input data in a variety of formats and converting it into a single format suitable for use with the store of map data;

User Service Requirements:

USR = 1.9.0;
USR = 1.9.1;
USR = 1.9.1.2.1;
USR = 1.9.1.2.2;
USR = 1.9.2.2.1;
USR = 1.9.2.2.3;

Output Flow Dynamics Assumptions:

tmup-request_pollution_display_update = 2/(60*60*24*7*52);
map_data_for_pollution_display = 2/(60*60*24*7*52);

1.5.4 Manage Pollution State Data Store

Input Flows

pollution_state
pollution_state_data_output_request
pollution_state_data_request
pollution_state_static_collection
pollution_state_vehicle_collection

Output Flows

archive_pollution_state_data
pollution_state
pollution_state_data
pollution_state_data_output

Description:

Overview: This process shall manage the store of pollution state data in the Manage Emissions facility of the Manage Traffic function. The data in the store shall be that which has been received by the process from other processes within the facility. The process shall manage the data in the store to enable its contents to be available to other processes within the Manage Traffic function, and to traffic operations personnel, via an interface process within the Manage Emissions facility.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'pollution_state_static_collection';
(b) 'pollution_state_data_request';
(c) 'pollution_state_data_output_request';
(d) 'pollution_state_vehicle_collection'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'pollution_state'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'pollution_state_data';
(b) 'pollution_state_data_output';
(c) 'archive_pollution_state_data'.

Functional Requirements: This process shall:
(a) run whenever any of the unsolicited input data flows shown above are received;
(b) when either 'pollution_state_static_collection' or 'pollution_state_vehicle_collection' of the unsolicited input flows is received, store the data that it contains in the store of pollution state data;
(c) when either the 'pollution_state_data_request' or the 'pollution_state_data_output_request' of the unsolicited input flows is received, read the contents of the store of pollution state data and send it to the requesting process using the appropriate solicited output flow shown above.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.2;
USR = 1.8.2.2(b);
USR = 1.9.0;
USR = 1.9.1;
USR = 1.9.1.2;
USR = 1.9.1.2.1;
USR = 1.9.1.2.2;
USR = 1.9.2.2.1;
USR = 1.9.2.2.3;

Output Flow Dynamics Assumptions:

pollution_state_data = 12/(60*60);
pollution_state_data_output = pollution_state_data_output_request;
archive_pollution_state_data = pollution_state_data;

1.5.5 Process Vehicle Pollution Data

Input Flows

From_Vehicle_Characteristics
ft_vehicle_pollutant_levels
pollution_state_vehicle_acceptance_criteria
vehicle_status_details_for_emissions

Output Flows

pollution_state_vehicle_collection
pollution_state_vehicle_log_data
vehicle_pollution_alert
vehicle_pollution_message_for_highways
vehicle_pollution_message_for_roads

Description:

Overview: This process shall obtain pollution data about individual vehicles and analyze it against reference data obtained from another process within the Manage Emissions facility of the Manage Traffic function. The process shall use this reference data to determine whether or not a vehicle is possibly violating the acceptable levels of pollution output. When the process determines that a possible violation has occurred, it shall send the detected pollution levels and the vehicle identity to the process responsible for law enforcement in the Manage Emergency Services function for action.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'From_Vehicle_Characteristics';
(b) 'ft-vehicle_pollutant_levels';
(c) 'pollution_state_vehicle_acceptance_criteria';
(d) 'vehicle_status_details_for_emissions'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'pollution_state_vehicle_log_data';
(b) 'pollution_state_vehicle_collection';
(c) 'vehicle_pollution_message_for_highways';
(d) 'vehicle_pollution_message_for_roads';
(e) 'vehicle_pollution_alert'.

Functional Requirements: This process shall:
(a) continuously monitor the unsolicited input data flows shown above;
(b) process the 'From_Vehicle_Characteristics' and 'ft-vehicle_pollutant_levels' unsolicited data flows data that shows the levels of various atmospheric pollutants (e.g. ozone and its precursors, carbon monoxide, nitrous oxide, sulfur dioxide, hydrocarbons, particles) being produced by a vehicle with a particular identity;
(c) use the data obtained from the 'pollution_state_vehicle_acceptance_criteria' unsolicited input data flow to determine whether or not the levels of the pollutants produced by the vehicle exceed acceptance criteria;
(d) continuously output current vehicle pollution levels to the process that manages the store of pollution state via the 'pollution_state_vehicle_collection' solicited output data flow as a locally determined aggregation for vehicles in a particular roadside location, including but not limited to removing the vehicle identity;
(e) if the pollution level(s) produced by a particular vehicle are found to exceed the acceptance criteria, send the 'vehicle_pollution_message_for_highways' or 'vehicle_pollution_message_for_roads' unsolicited data flow to the Provide Device Control facility so that the vehicle's driver can be given visual warning of what the vehicle is doing to the atmospheric pollution levels, and send the 'vehicle_pollution_alert' data flow to the Manage Emergency Service function for enforcement purposes;
(f) data concerning the levels of pollution being detected in vehicles shall be sent to the store that is acting as the log of pollution data using the 'pollution_state_vehicle_log_data' solicited output flow, having had all vehicle identity data removed, but vehicle type data retained.

User Service Requirements:

USR = 1.0;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.4;
USR = 1.8.1.4(b);
USR = 1.8.2;

USR = 1.8.2.13;
USR = 1.8.2.13(b);
USR = 1.8.3;
USR = 1.8.3.1;
USR = 1.8.3.1(d);
USR = 1.9;
USR = 1.9.0;
USR = 1.9.2;
USR = 1.9.2.1;
USR = 1.9.2.1.1;
USR = 1.9.2.1.2;
USR = 1.9.2.1.4;
USR = 1.9.2.1.5;
USR = 1.9.2.2;
USR = 1.9.2.2.1;
USR = 1.9.2.2.2;
USR = 1.9.2.2.3;

Output Flow Dynamics Assumptions:

pollution_state_vehicle_log_data = 12/(60*60);
pollution_state_vehicle_collection = 1;
vehicle_pollution_message_for_highways = (1/(60*60*24*7*52)*VEHS)*1000;
vehicle_pollution_message_for_roads = (1/(60*60*24*7*52)*VEHS)*1000;
vehicle_pollution_alert = (1/(60*60*24*7*52)*VEHS)*1000;

1.5.6 Detect Roadside Pollution Levels

Input Flows

fe_roadside_pollutant_levels

Output Flows

pollution_state_roadside_collection

Description:

Overview: This process shall process the local area pollution data analyzed by sensors looking at the levels of pollution at the roadside within the geographic area served by the Manage Traffic function. The process shall pass the data on to another process within the Manage Emissions facility for integration with wide area pollution data and comparison with thresholds for pollution incidents.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fe-area_pollutant_levels';

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'pollution_state_roadside_collection'.

Functional Requirements: This process shall:
(a) monitor the unsolicited input data flow 'fe-area_pollutant_levels';
(b) process the 'fe-area_pollutant_levels' unsolicited data flow into data that shows the levels of various atmospheric pollutants(e.g. ozone and its precursors, carbon monoxide, nitrous oxide, sulfur dioxide, hydrocarbons, particulates);
(c) output measures of the current pollution levels to the process that compares them with the threshold values for pollution incidents.

User Service Requirements:

USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.4;
USR = 1.8.1.4(d);
USR = 1.9.0;
USR = 1.9.1;
USR = 1.9.1.1;
USR = 1.9.1.1.1;
USR = 1.9.1.1.2;
USR = 1.9.1.1.3;
USR = 1.9.1.2;
USR = 1.9.1.2.1;
USR = 1.9.1.2.2;

Output Flow Dynamics Assumptions:

pollution_state_roadside_collection = 1;

1.5.7 Manage Pollution Data Log

Input Flows

pollution_data_log
pollution_state_static_log_data
pollution_state_vehicle_log_data

Output Flows

pollution_archive_data_log
pollution_data_log

Description:

Overview: This process shall manage the log of pollution data within the Manage Emissions facility of the Manage Traffic function. The process shall receive data for entry into the log from other processes within the facility. It shall also send the contents of the log to the Manage Archive Data function for use in planning future modifications to the ITS network.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'pollution_state_static_log_data';
(b) 'pollution_state_vehicle_log_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'pollution_data_log'.

Unsolicited Output Processing: This process shall provide the following output flow regardless of whether or not any of the above inputs have been received:
(a) 'pollution_archive_data_log'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows 'pollution_state_static_log_data' and 'pollution_state_vehicle_log_data';
(b) when either of the unsolicited flows in (a) is received, load the data that it contains into the store of pollution log data 'pollution_data_log'.

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

pollution_data_log = pollution_state_static_log_data + pollution_state_vehicle_log_data;
pollution_archive_data_log = pollution_state_static_log_data + pollution_state_vehicle_log_data;

1.5.8 Manage Pollution Reference Data Store

Input Flows

pollution_reference_data
pollution_reference_data_archive_request
pollution_reference_data_request
pollution_reference_data_update

Output Flows

archive_pollution_reference_data
pollution_reference_data
pollution_reference_data_output
pollution_state_static_acceptance_criteria
pollution_state_vehicle_acceptance_criteria

Description:

Overview: This process shall manage the store of pollution reference data within the Manage Emissions facility of the Manage Traffic function. It shall make the contents of the store available to other processes within the facility that are responsible for emissions management, and on request to the traffic operations personnel interface process. The process shall accept updates to the stored data from the traffic operations personnel interface process.

(a) 'pollution_reference_data_request';
(b) 'pollution_reference_data_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'pollution_reference_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'pollution_reference_data_output';
(b) 'pollution_state_static_acceptance_criteria';
(c) 'pollution_state_vehicle_acceptance_criteria';
(d) 'archive_pollution_reference_data'.

Functional Requirements: This process shall:
(a) run whenever either the 'pollution_reference_data_request' or 'pollution_reference_data_update' unsolicited input data flows are received;
(b) when the 'pollution_reference_data_request' unsolicited input flow is received, read the contents of the store of pollution reference data and send 'pollution_reference_data_output' to the requesting process;
(c) when the 'pollution_reference_data_update' unsolicited input flow is received, store the data that it contains in the store of pollution state data, if necessary over writing data already present;
(d) when (c) is complete, output the received data to the appropriate process using either the 'pollution_state_static_acceptance_criteria' or 'pollution_state_vehicle_acceptance_criteria' solicited output flows;

User Service Requirements:

USR = 1.9.0;
USR = 1.9.1;
USR = 1.9.1.2;
USR = 1.9.1.2.1;
USR = 1.9.1.2.2;
USR = 1.9.2.2.1;
USR = 1.9.2.2.3;

Output Flow Dynamics Assumptions:

pollution_state_static_acceptance_criteria = pollution_reference_data_update;
pollution_reference_data = pollution_reference_data_update;
pollution_reference_data_output = pollution_reference_data_request;
pollution_state_vehicle_acceptance_criteria = pollution_reference_data_update;
archive_pollution_reference_data = pollution_reference_data;

1.5.9 Manage Pollution Archive Data

Input Flows

archive_pollution_data
archive_pollution_reference_data
archive_pollution_state_data
emissions_archive_request
emissions_archive_status
emissions_data_archive
pollution_archive_data_log

Output Flows

emissions_archive_data
emissions_data_archive
pollution_reference_data_archive_request

Description:

Overview: This process shall collect and store the pollution data being collected from sensors in the geographic area being served by the Manage Traffic function. The process shall integrate data from distributed roadside sensors (provided by another process) with that obtained directly from sensors looking at the general (wide area) environment.

Data Flows: All input data flows from the are unsolicited with the exception of emissions_archive_status and all output flows which are solicited.

Functional Requirements: This process shall:
(a)continuously monitor receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited inputs shown above are received, the process shall immediately generate the solicited output shown above;
(c) data shall only be sent to the source from which the data request originated.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.7;
USR = 7.1.3.1.7(a);

Output Flow Dynamics Assumptions:

emissions_archive_data = emissions_archive_data_request;
pollution_reference_data_archive_request = archive_pollution_reference_data;

1.6.1.1 Detect Roadway Events

Input Flows

approaching_train_data
device_control_state
hri_device_sense
hri_status
hri_traffic_surveillance
indicator_sign_control_data_for_hri

Output Flows

current_hri_state
event_notice
roadway_status
train_sense_data

Description:

Overview: This process is responsible for monitoring local sensor data obtained from traffic surveillance and then determining and reporting the current state of all traffic in the HRI vicinity. The process provides triggers for other processes within Manage HRI Traffic Volume. It also monitors the device controls as they are initiated by the Activate HRI Device Controls process.

User Service Requirements:

USR = 1.0;
USR = 1.3;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.2.1(d);
USR = 1.3.1.2.1(d).1;
USR = 1.10;
USR = 1.10.0;
USR = 1.10.1;
USR = 1.10.1.7;
USR = 1.10.2;
USR = 1.10.2.1;
USR = 1.10.2.1.1;
USR = 1.10.3;
USR = 1.10.3.3;
USR = 1.10.3.3.3;

Output Flow Dynamics Assumptions:

roadway_status = device_control_state + approaching_train_data;
current_hri_state = 1;
event_notice = approaching_train_data;
train_sense_data = 1;

1.6.1.2.1 Control HRI Traffic Signals

Input Flows

hri_control_message

Output Flows

barrier_control_request
hsr_control_request
ssr_control_request
traffic_device_control
traffic_device_control_state

Description:

Overview: This process is responsible for interpreting the hri_control message and safely directing the activation of the appropriate devices. This process will both directly command devices at the HRI and will disseminate necessary control information to the Process Indicator Output Data for Roads function to allow integrated control of adjacent traffic signals. Data will also be sent to SSR and/or HSR Device Control functions to control these specialized devices at the crossing. When sensor data indicates an approaching train this process notifies the Process Indicator Output Data for Roads function to allow the signal timing to be adjusted and dynamic message signs, if available, to be updated. This allows the traffic signals in the area adjacent to an HRI to be used to clear the Storage Area in advance of an approaching train and to manage traffic around the intersection.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.1;
USR = 1.10.1.7;
USR = 1.10.3;
USR = 1.10.3.1;
USR = 1.10.3.2;
USR = 1.10.3.3;
USR = 1.10.3.3.1;

Output Flow Dynamics Assumptions:

ssr_control_request = hri_control_message;
traffic_device_control_state = hri_control_message;
hsr_control_request = hri_control_message;
barrier_control_request = hri_control_message;
traffic_device_control = hri_control_message;

1.6.1.2.2 Control HRI Warnings and Barriers

Input Flows

barrier_control_request

Output Flows

barrier_device_control
barrier_device_control_state

Description:

Overview: This process is responsible for initiating the activation of HRI barriers at active vehicular and pedestrian grade crossings. When a request is sent to activate the HRI barriers perhaps because of a detection of an oncoming train, this process sends the device control signal to the Manage Device Controls process to activate the barriers. This process also returns state information to the Maintain Device State process concerning the commands that have been initiated by this process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.3;
USR = 1.10.3.3.2;

Output Flow Dynamics Assumptions:

barrier_device_control_state = barrier_control_request;
barrier_device_control = barrier_control_request;

1.6.1.2.3 Provide SSR Device Controls

Input Flows

ssr_control_request

Output Flows

ssr_device_control
ssr_device_control_state

Description:

Overview: This process is responsible for initiating the activation of HRI Standard Speed Rail control devices at active vehicular and pedestrian grade crossings. This process responds to requests sent by the Control HRI Traffic Signals process based on detection of an oncoming train. This process sends command information to the Manage Device Control containing control signals and commands that are unique to the SSR functions. State information is also sent to the Maintain Device State process to monitor the last known state of the controls commands being processed.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.4;
USR = 1.10.4.1;

Output Flow Dynamics Assumptions:

ssr_device_control_state = ssr_control_request;
ssr_device_control = ssr_control_request;

1.6.1.2.4 Provide HSR Device Controls

Input Flows

hsr_control_request

Output Flows

hsr_device_control
hsr_device_control_state

Description:

Overview: This process is responsible for initiating the activation of HRI devices, barriers and other special safety features for High Speed Rail at active vehicular and pedestrian grade crossings. This process responds to requests sent by the Control HRI Traffic Signals process based on detection of an oncoming train. This process sends command information to the Manage Device Control containing control signals and commands that are unique to the HSR functions, such as trapped vehicle detection. State information is also sent to the Maintain Device State process to monitor the last known state of the controls commands being processed.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.5;
USR = 1.10.5.1;
USR = 1.10.5.2;
USR = 1.10.5.2.2;

Output Flow Dynamics Assumptions:

hsr_device_control_state = hsr_control_request;
hsr_device_control = hsr_control_request;

1.6.1.2.5 Manage Device Control

Input Flows

barrier_device_control
hsr_device_control
ssr_device_control
traffic_device_control

Output Flows

hri_device_control

Description:

Overview: This process is responsible for managing and selecting the appropriate device control messages. This process gathers the control signals from the other Activate HRI Device Control processes and forwards them as needed to the Process Indicator Output Data for Roads process within Provide Device Control. These control signals are used to activate all of the HRI unique roadside devices such as gates or other barriers, lights, adjacent traffic signals, message signs or in-vehicle signage beacons.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.4;
USR = 1.10.4.1;

Output Flow Dynamics Assumptions:

hri_device_control = ssr_device_control + hsr_device_control + traffic_device_control + barrier_device_control;

1.6.1.2.6 Maintain Device State

Input Flows

barrier_device_control_state
hsr_device_control_state
ssr_device_control_state
traffic_device_control_state

Output Flows

device_control_state

Description:

Overview: This process is responsible for managing and selecting the appropriate device control state messages. This process collects the device state messages that are produced by the other Activate HRI Device Controls processes and forwards the appropriate signals to the Detect Roadway Events process that monitors the status of the HRI commands being processed. This information is also used in the equipment diagnostic monitoring and testing.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.4;
USR = 1.10.4.1;

Output Flow Dynamics Assumptions:

device_control_state = ssr_device_control_state + hsr_device_control_state + traffic_device_control_state + barrier_device_control_state;

1.6.1.3 Perform Equipment Self-Test

Input Flows

hri_device_sense
near_term_status

Output Flows

device_status

Description:

Overview: This process is responsible for performing real-time equipment checks and reporting the status of the equipment associated with an active grade crossing. Based on receipt of the sensor data of the surrounding highway and rail traffic and receipt of any near term events this process can execute a real-time check of the equipment and determine the relative health and status of the active grade crossing equipment. The output is sent onto the Monitor HRI Status process for further processing with other diagnostic data.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.3;
USR = 1.10.3.3.4;

Output Flow Dynamics Assumptions:

device_status = hri_device_sense + near_term_status;

1.6.1.4.1 Generate Alerts and Advisories

Input Flows

hazard_condition

Output Flows

hri_advisory
hri_alert

Description:

Overview: This process is responsible for generating the messages to advise and protect motorists, travelers and train crews approaching and crossing railroad grade crossings. Based on the severity of the hazard condition sent by the Detect HRI Hazards process this process will either send an hri_advisory command for non-time critical data or an hri_alert command for time critical data to the Report Alerts and Advisories. These users that will receive these messages include drivers, bicyclists, and pedestrians.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.1;
USR = 1.10.1.5;

Output Flow Dynamics Assumptions:

hri_advisory = hazard_condition;
hri_alert = hazard_condition;

1.6.1.4.2 Provide Closure Parameters

Input Flows

hazard_condition

Output Flows

time_to_closing

Description:

Overview: This process is responsible for providing the HRI predicted time to closure to be used in broadcast message alerts to approaching vehicles. This time is calculated from data provided by the Detect HRI Hazards process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.2;
USR = 1.10.2.2.4;

Output Flow Dynamics Assumptions:

time_to_closing = hazard_condition;

1.6.1.4.3 Report Alerts and Advisories

Input Flows

hri_advisory
hri_alert

Output Flows

approach_warning
hri_guidance_for_dms
train_message

Description:

Overview: This process is responsible for reporting real-time HRI traffic volume advisories and real-time highway traffic alerts. Depending on the input received from the Generate Alerts and Advisories process, this process sends alerts or advisories to a train to describe the operational status of the intersection and alerts about any hazards. This process also sends the commands to Output Control Data for Roads process that will control the dynamic message signs in the area of an HRI to display the appropriate alert or advisory. Messages for local beacon broadcast are processed and sent to the Report HRI Status on Approach process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.3;
USR = 1.10.3.3.5;

Output Flow Dynamics Assumptions:

train_message = hri_alert;
hri_guidance_for_dms = hri_advisory;
approach_warning = hri_alert;

1.6.1.4.4 Report HRI Status on Approach

Input Flows

approach_warning
hazard_condition
time_to_closing

Output Flows

hri_guidance_for_beacon_message

Description:

Overview: This process is responsible for providing real-time HRI status to vehicles as they approach an HRI. It must discriminate between vehicles near, but not approaching, the HRI (e.g. on parallel side streets, etc.). This process develops the message to be broadcast to nearby vehicles by receiving time_to_closing data and the hazard_condition signal and calculating the appropriate window of time to display the message. The message is built from the approach_warning data received from the Report Alerts and Advisories process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.5;
USR = 1.10.5.2;
USR = 1.10.5.2.6;

Output Flow Dynamics Assumptions:

hri_guidance_for_beacon_message = hazard_condition;

1.6.1.5 Detect HRI Hazards

Input Flows

hri_hazard

Output Flows

hazard_condition
intersection_blocked
strategy_preemption

Description:

Overview: This process is responsible for detecting real-time HRI blockages or collisions in the vicinity of an HRI that create a blockage or other hazard at the HRI. Based upon information received from the Provide Advance Warnings process this process can send a request to the Control Traffic Volume at Active HRI that the local signal strategy be preempted. A hazard condition message can also be sent to the Generate Alerts and Advisories process for further action or the Provide Closures Parameters process to possibly adjust the time to closing.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.3;
USR = 1.10.3.3.3;
USR = 1.10.6;

Output Flow Dynamics Assumptions:

strategy_preemption = hri_hazard;
hazard_condition = hri_hazard;
intersection_blocked = hri_hazard;

1.6.1.6.1 Close HRI on Detection

Input Flows

current_hri_state
hri_predicted_collision
local_control_plan
rail_operations_advisories

Output Flows

hri_blockage
hri_hazard
near_term_status
predicted_hri_state
rail_operations_message

Description:

Overview: This process is responsible for protecting highway vehicles approaching and crossing railroad grade crossings by initiating the closure up to 3 minutes before train arrival. This process receives the near term status of the crossing including any approaching trains or trapped vehicles. With this information along with the local control plan data the predicted hri state is computed and sent to the Detect Imminent Vehicle/Train Collision process. If a hri_predicted_collision message is returned then this process sends out an hri_hazard message to the Detect HRI Hazard which will in turn result in a change to the device control strategy. This process also receives rail operations advisories for processing along with the state and control plan data. As needed this process will output any rail_operations_message data to the Interact with Rail Operations process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.1;
USR = 1.10.1.4;
USR = 1.10.5;
USR = 1.10.5.2;
USR = 1.10.5.2.1;

Output Flow Dynamics Assumptions:

predicted_hri_state = current_hri_state;
hri_hazard = current_hri_state;
near_term_status = current_hri_state;
rail_operations_message = current_hri_state;
hri_blockage = current_hri_state;

1.6.1.6.2 Detect Imminent Vehicle/Train Collision

Input Flows

predicted_hri_state

Output Flows

hri_predicted_collision

Description:

Overview: This process is responsible for detecting imminent collisions between vehicles and trains at railroad grade crossings. Using the data contained in the predicted_hri_state message this process performs the necessary calculations to determine whether a collision is imminent. If so, this process returns a hri_predicted_collision message to the Close_HRI_on_Detection process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.1;

Output Flow Dynamics Assumptions:

hri_predicted_collision = predicted_hri_state;

1.6.1.7.1 Control Vehicle Traffic at Passive HRI

Input Flows

rail_operations_device_command

Output Flows

crew_close_hri

Description:

Overview: This process is responsible for controlling traffic volume at passive grade crossings. It provides a mechanism for rail operations to close grade crossings that have active traffic devices but no real-time train detection mechanisms. This process also will allow for a train crew member to manually activate closure of the crossing. In such an event a crew_close_hri signal is sent to the Close_HRI_on_Command process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.4;
USR = 1.10.4.2;
USR = 1.10.4.2.1;

Output Flow Dynamics Assumptions:

crew_close_hri = 1/MONTH;

1.6.1.7.2 Control Vehicle Traffic at Active HRI

Input Flows

event_notice
hri_traffic_surveillance
preemption_command
strategy_preemption

Output Flows

close_hri
hri_traffic_data
local_control_plan
traffic_management_request

Description:

Overview: This process is responsible for controlling vehicular traffic at an active HRI by controlling the operation of traffic control devices in accordance with a predetermined local control plan. The local_control_plan is communicated to the Close_HRI_on_Detection process. This local control plan can be preempted by a strategy_preemption message from the Detect_HRI_Hazards process or by such inputs as an event_notice from the Detect_Roadway_Events process or hri_traffic_surveillance data. The outputs of this process include the command messages to close the HRI, requests for information from the Manage Traffic function, and information about the current hri_traffic_data.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;

Output Flow Dynamics Assumptions:

close_hri = event_notice;
traffic_management_request = 1/HOUR;
hri_traffic_data = event_notice;
local_control_plan = 1/DAY;

1.6.1.7.3 Close HRI on Command

Input Flows

close_hri
crew_close_hri
rail_operations_device_command

Output Flows

hri_control_message

Description:

Overview: This process is responsible for closing the HRI to vehicular traffic, either on command from the Control Traffic Volume at Active HRI process, or from direct command from rail operations (as an override). A third mechanism for closing the HRI is defined for passive crossings, i.e. crossings without active train detection systems. Upon command from rail operations, or via manual operation by a train crewman, active traffic devices at an otherwise passive grade crossing may be activated to close the crossing.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.4;
USR = 1.10.4.1;
USR = 1.10.5;
USR = 1.10.5.2;
USR = 1.10.5.2.1;

Output Flow Dynamics Assumptions:

hri_control_message = rail_operations_device_command + close_hri + crew_close_hri ;

1.6.2.1 Exchange Data with Rail Operations

Input Flows

fro_incident_notification
fro_maintenance_schedules
fro_train_schedules
hri_priority_message
rail_operations_message

Output Flows

rail_operations_device_command
rail_operations_priority_data
rail_operations_update
ro_requests
tro_equipment_status
tro_event_schedules
tro_incident_notification

Description:

Overview: This process is responsible for exchanging routine data with rail operations. Such data being sent to the rail operators includes event schedules, requests for information from the Rail Operators, incident notification based on rail operations messages received from Close_HRI_on_Detection process and hri_priority_message data received from the Manage Alerts and Advisories process. This process receives maintenance schedules, train schedules, and incident notifications from the rail operators. This information is used to develop the rail operations update data that is passed onto the Manage Rail Traffic Control Data process and the rail operations priority data that is sent to the Manage Alerts and Advisories process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.1;

Output Flow Dynamics Assumptions:

tro-event_schedules = 1/DAY;
rail_operations_priority_data = 1/DAY;
rail_operations_update = 1/DAY;
rail_operations_device_command = 1/DAY;
tro-incident_notification = 1/DAY;
ro_requests = 1/DAY;
tro_equipment_status = 1/DAY;

1.6.2.2 Manage Alerts and Advisories

Input Flows

hri_blockage
hri_status
rail_operations_data
rail_operations_priority_data

Output Flows

hri_priority_message
rail_operations_advisories
rail_operations_query

Description:

Overview: This process is responsible for acquiring HRI advisory or alert data from rail operations and for providing HRI status to rail operations. The data managed by this process may be time critical, as in the case of alerts or priority messages, or not time critical, as in the case of advisories.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.2;
USR = 1.10.2.2.1;
USR = 1.10.5;
USR = 1.10.5.2;
USR = 1.10.5.2.4;

Output Flow Dynamics Assumptions:

rail_operations_query = 1/DAY;
hri_priority_message = 1/DAY;
rail_operations_advisories = 1/DAY;

1.6.2.3 Manage Rail Traffic Control Data

Input Flows

rail_operations_query
rail_operations_update
rail_traffic_control_data
request_rail_schedules_data

Output Flows

rail_operations_data
rail_schedules_data
rail_traffic_control_data

Description:

Overview: This process is responsible for providing and maintaining a current store of rail operations data. The data is assembled from the rail_operations_update information sent by the Exchange Data with Rail Operations process. Queries for this information are received from the Manage Alerts and Advisories process and the Interact with Vehicle Traffic Management processes.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.2;
USR = 1.10.2.2.2;

Output Flow Dynamics Assumptions:

rail_traffic_control_data = rail_operations_update + rail_operations_query;
rail_operations_data = rail_operations_query;
rail_schedules_data = request_rail_schedules_data;

1.6.3.1 Interact with Wayside Systems

Input Flows

ats_alert
fwe_approaching_train_announcement
fwe_train_data
fwe_wayside_equipment_status
hri_reporting_data

Output Flows

approaching_train_announcement
approaching_train_data
ats_status
twe_hri_status
twe_stop_highway_indication
twe_stop_train_indication
wayside_status

Description:

Overview: This process is responsible for interfacing to railroad owned and maintained wayside equipment, such as Wayside Interface Units, Crossing Gate Controllers, etc. All these devices are expected to provide real-time information to the HRI about approaching trains and their own health. In addition, advanced implementations will make use of a communications path back to approaching trains provided by the railroad's equipment.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.1;
USR = 1.10.1.1;
USR = 1.10.1.2;
USR = 1.10.1.3;
USR = 1.10.1.6;
USR = 1.10.1.7;
USR = 1.10.2;
USR = 1.10.2.2;

Output Flow Dynamics Assumptions:

twe-stop_highway_indication = fwe-approaching_train_announcement;
twe-stop_train_indication = fwe-approaching_train_announcement;
ats_status = 1/HOUR;
approaching_train_announcement = fwe-approaching_train_announcement;
wayside_status = 1/HOUR;
approaching_train_data = fwe-approaching_train_announcement;
twe-hri_status = fwe-approaching_train_announcement;

1.6.3.2 Advise and Protect Train Crews

Input Flows

approaching_train_announcement
ats_warning_notification
hri_status
train_message

Output Flows

ats_advisory
hri_reporting_data

Description:

Overview: This process is responsible for generating advisories/ alerts that are routed to the wayside equipment for transmission to the train crews. If the intersection is blocked, or there is an incident at the intersection this information will be passed to the Interact with Wayside Systems process for routing to the wayside equipment. The wayside equipment can then route the information directly to the train crews, or to rail operations.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.1;
USR = 1.10.1.6;

Output Flow Dynamics Assumptions:

ats_advisory = approaching_train_announcement;
hri_reporting_data = approaching_train_announcement;

1.6.3.3 Provide ATS Alerts

Input Flows

ats_advisory
ats_status

Output Flows

ats_alert
ats_warning_notification
hri_rail_alert

Description:

Overview: This process is responsible for automatically protecting commuter, intercity, transit and freight trains as they approach and cross grade crossings. It also reports HRI rail traffic advisories to traffic management and rail operations. It is responsible for verifying and reporting overall HRI status to approaching trains so that crews can act within safe service braking distances. It provides for notification of Automatic Train Stop systems (ATS, PTS, etc) with sufficient advance warning to allow emergency brake application time to stop a train before it encounters an HRI hazard. Finally, it provides automatic status indications about the HRI to the crews of approaching trains.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.1;
USR = 1.10.1.2;
USR = 1.10.1.3;
USR = 1.10.3;
USR = 1.10.3.3;
USR = 1.10.3.3.5;
USR = 1.10.5;
USR = 1.10.5.2;
USR = 1.10.5.2.3;
USR = 1.10.5.2.5;

Output Flow Dynamics Assumptions:

ats_alert = ats_status;
ats_warning_notification = ats_status;
hri_rail_alert = ats_status;

1.6.4.1 Manage HRI Closures

Input Flows

hri_strategy_override
hri_traffic_data
train_ops_plan

Output Flows

closure_event_data
hri_incident_data

Description:

Overview: This process is responsible for coordination and managing of HRI closures at the Traffic management
Center. It interfaces with Manage Incidents process to provide incident information and to receive strategy

overrides as required by the larger incident management function.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.1;
USR = 1.10.2.1.3;

Output Flow Dynamics Assumptions:

closure_event_data = hri_traffic_data;
hri_incident_data = hri_strategy_override;

1.6.4.2 Exchange Data with Traffic Management

Input Flows

closure_event_data
hri_status
intersection_blocked
rail_schedules_data
traffic_management_request
traffic_surveillance_data

Output Flows

hri_sensor_data
hri_status_for_traffic_demand
hri_traffic_surveillance
request_rail_schedules_data
tms_requests
train_ops_plan

Description:

Overview: This process is responsible for interacting with traffic management processes. It collects data from processes that are within the HRI elements located at the roadside and forwards the data as needed to other processes within traffic management. It also acts as the interface between rail operations and traffic management processes through its interface with the Interact with Rail Operations process.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.1;
USR = 1.10.2.1.2;
USR = 1.10.2.2;
USR = 1.10.2.2.4;

Output Flow Dynamics Assumptions:

hri_status_for_traffic_demand = 1/HOUR;
hri_traffic_surveillance = 1/HOUR;
train_ops_plan = 1/HOUR;
tms_requests = 1/HOUR;
request_rail_schedules_data = 1/HOUR;
hri_sensor_data = 1;

1.6.5.1 Provide Interactive Interface

Input Flows

hri_closure_data_response
hri_state
ro_requests
tms_requests

Output Flows

hri_status
request_hri_closure_data

Description:

Overview: This process is responsible for initiating reports of the health status of the HRI to both Traffic Management and Rail Operations. In addition the process initiates reporting of the health status of the HRI to the wayside interface equipment (and ultimately to the train when the advanced HRI functionality is in place).

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.2;
USR = 1.10.2.2.1;
USR = 1.10.2.2.2;
USR = 1.10.2.2.3;

Output Flow Dynamics Assumptions:

request_hri_closure_data = ro_requests + tms_requests;
hri_status = ro_requests + tms_requests;

1.6.5.2 Determine HRI Status

Input Flows

device_status
hri_rail_alert
roadway_status
wayside_status

Output Flows

hri_state
preemption_command

Description:

Overview: This process is responsible for monitoring critical HRI functions and merging them into a single coherent picture of the state of the hri. It also is responsible for assuring that the HRI always reverts to the safest possible operating condition in the event of any operational malfunctions.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.3;
USR = 1.10.3.1;

Output Flow Dynamics Assumptions:

hri_state = roadway_status + device_status + wayside_status + hri_rail_alert;
preemption_command = 1/DAY;

1.6.5.3 Maintain HRI Closure Data

Input Flows

hri_closure_data
hri_state
request_hri_closure_data

Output Flows

hri_closure_data
hri_closure_data_response

Description:

Overview: This process is responsible for managing a log of the HRI operation for use in strategy planning, demand management and traffic management.

User Service Requirements:

USR = 1.0;
USR = 1.10;
USR = 1.10.2;
USR = 1.10.2.1;
USR = 1.10.2.1.3;

Output Flow Dynamics Assumptions:

hri_closure_data = request_hri_closure_data + hri_state;
hri_closure_data_response = request_hri_closure_data;

2.1.1 Manage Commercial Fleet Electronic Credentials and Tax Filing

Input Flows

cf_enrollment_information
cf_enrollment_payment_confirmation
cf_hazmat_request
cf_manager_activity_report_request
cf_manager_enrollment_payment_request
cf_manager_enrollment_request
cf_manager_route_request
cf_manager_storage_request
cf_periodic_activity_report
cf_retained_data
cf_roadside_activity_report
cf_route
cf_route_details
cf_static_route_data
cf_tag_data
freight_cargo_data

Output Flows

cf_driver_route
cf_enrollment_payment_request
cf_enrollment_request
cf_hazmat_route_information
cf_hazmat_vehicle_information
cf_manager_activity_report
cf_manager_enrollment_information
cf_manager_enrollment_payment_confirmation
cf_manager_route_data
cf_request_activity_report
cf_retained_data
cf_route_details
cf_route_request
cf_static_route_request
cf_tag_initialization_data
cf_tax_audit_data

Description:

Overview: This process shall be responsible for providing the commercial vehicle manager with the ability to manage the activities of commercial vehicles. The process shall provide the capability for the manager to obtain commercial vehicle routes. When a route has been confirmed, the process shall enable the manager to enroll commercial vehicles for electronic clearance at roadside check station facilities, to process and pay for electronic credential and tax filing, to send tag data to the Provide Commercial Vehicle On-board Data facility, and to provide vehicle route instructions for use by the commercial vehicle driver. Periodically it shall also send reports about taxes that have been paid to the Administer Commercial Vehicles facility. The process shall also enable the manager to obtain commercial vehicle activity reports from the logs provided by roadside checkstation facilities. It shall be possible to obtain these reports either on request or at periodic intervals.

Data Flows: All input data flows are unsolicited and all output data flows are solicited with the exception of the following:
(a) 'cf_route_details', which contains data requested from and written to a data store;
(b) 'cf_driver_route', which contains data written to a data store;
(c) 'cf_enrollment_information', 'cf_enrollment_payment_confirmation', 'cf_route' and 'cf_static_route_data', which are received as the result of output to other processes.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received, generate the appropriate outputs identified below:
(c) if the input in (b) contains data to set up a route for a commercial vehicle, use the input data, plus any details of previously used routes in the local store to construct a route request and send it to either the static route selection process, or to the Provide Driver and Traveler Services function for a dynamic route, i.e. one that takes account of current and future traffic conditions;
(d) when the route requested in (c) has been provided, or the input requests that the electronic credentials be obtained and the tax details filed, send the data to the commercial vehicle administration process;
(e) repeat (d) to enable payment to be made;

(f) when the route has been determined and the electronic credentials and tax filing have been obtained and paid for, load the route details into the data store for access by the commercial vehicle driver;
(g) periodically send details of the taxes that have been paid to the Administer Commercial Vehicles function;
(h) the process shall be responsible for the management of the data in the stores of commercial fleet route details and retained data, and for writing to the store of driver instructions, using the most appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.4;
USR = 4.4.1;
USR = 4.4.1(e);
USR = 4.4.1(f);
USR = 4.4.1(g);
USR = 4.6;
USR = 4.6.1;

Output Flow Dynamics Assumptions:

cf_driver_route = cf_manager_route_request;
cf_enrollment_request = cf_manager_enrollment_request;
cf_enrollment_payment_request = cf_manager_enrollment_payment_request;
cf_hazmat_route_information = cf_hazmat_request;
cf_hazmat_vehicle_information = cf_hazmat_request;
cf_manager_activity_report = cf_manager_activity_report_request;
cf_manager_enrollment_payment_confirmation = cf_manager_enrollment_payment_request;
cf_manager_enrollment_information = cf_manager_enrollment_request;

2.1.2 Provide Commercial Fleet Static Route

Input Flows

cf_static_route_request
map_data_for_fleet_managers

Output Flows

cf_static_route_data
map_data_for_fleet_managers

Description:

Overview: This process shall be responsible for providing a static commercial vehicle route using data provided by the commercial vehicle manager. A static route is one which is based on geographic data and therefore takes no account of current or predicted traffic conditions, incidents, etc. The process shall provide the route using its own route generation algorithms and data from its own store of digitized map information.

Data Flows: The input data flow is unsolicited and the output flow is solicited. The following data flow contains data requested from and written to a data store:
(a) 'map_data_for_fleet_managers'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the requested route and send its details in the output identified above;
(c) be responsible for the management of the data in the store of static data, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.4.0;
USR = 4.4.1;
USR = 4.4.1(c);
USR = 4.4.1(d);

Output Flow Dynamics Assumptions:

cf_static_route_data = 2/(60*60*24*7)*CVO_VEHS;

2.1.3 Provide Flt Mgr Electronic Credentials and Tax Filing Interface

Input Flows

cf_driver_route_instructions_output
cf_manager_activity_report
cf_manager_credit_identity
cf_manager_enrollment_information
cf_manager_enrollment_payment_confirmation
cf_manager_route_data
cf_vehicle_data
fcvm_enrollment_payment_request
fcvm_enrollment_request
fcvm_other_data_input
fcvm_preclearance_data
fcvm_request_driver_route_instructions
fcvm_request_on_board_vehicle_data
fcvm_roadside_activity_report_request
fcvm_route_data
fcvm_route_function_request
fcvm_update_driver_route_instructions

Output Flows

cf_driver_instructions_request
cf_driver_load_data
cf_manager_activity_report_request
cf_manager_enrollment_cost
cf_manager_enrollment_payment_request
cf_manager_enrollment_request
cf_manager_route_request
cf_manager_storage_request
cf_request_vehicle_data
tcvm_data_input_request
tcvm_driver_route_instructions
tcvm_enrollment_confirmation
tcvm_enrollment_payment_confirmation
tcvm_other_data_request
tcvm_preclearance_results
tcvm_roadside_activity_report
tcvm_route_data

Description:

Overview: This process shall be responsible for providing an interface for the commercial vehicle manager. The process shall enable this interface to provide the manager with facilities for the input of data used to set up commercial vehicle routes, to pay the necessary taxes and duties so that a commercial vehicle can be enrolled for a particular route, to exchange general information messages with a driver in a vehicle, and to set up instructions for a driver to take a vehicle on a particular route. It shall be possible for the driver's route instructions input by the manager to include details of the cargo to be picked up and/or dropped off at each point along the route. The enrollment activity supported by the process shall enable a commercial vehicle to pass through the roadside checkstations along its route without stopping, unless safety checks are required. The process shall support inputs from the commercial vehicle manager in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be in hardcopy, or as a display. output flows are solicited. The other input flows are as a result of sending outputs to other processes.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the commercial vehicle manager terminator listed above;
(b) when the inputs are received, generate the appropriate output flows identified above to other processes and monitor for the responses that they produce;
(c) when the responses in (b) are received, generate the appropriate output flows to the commercial vehicle manager terminator, using the flows listed above..

User Service Requirements:

USR = 4.0;
USR = 4.6;
USR = 4.6.1;
USR = 4.6.2;

Output Flow Dynamics Assumptions:

cf_driver_load_data = fcvm-update_driver_route_instructions;
cf_driver_instructions_request = fcvm-request_driver_route_instructions;
cf_manager_activity_report_request = fcvm-roadside_activity_report_request;
cf_manager_storage_request = 1/(60*60*24*7)*CVO_VEHS;
cf_manager_route_request = 1/(60*60*24)*CVO_VEHS;
cf_manager_enrollment_cost = 4/(60*60*24*7)*CVO_VEHS;
cf_manager_enrollment_request = 4/(60*60*24*7)*CVO_VEHS;
cf_manager_enrollment_payment_request = 4/(60*60*24*7)*CVO_VEHS;
cf_request_vehicle_data = fcvm-request_on_board_vehicle_data;
tcvm-data_input_request = 4/(60*60)*CVO_MAN;
tcvm-driver_route_instructions = fcvm-request_driver_route_instructions;
tcvm-enrollment_confirmation = 4/(60*60*24*7)*CVO_VEHS;

2.1.4 Provide Fleet Manager Commercial Vehicle Communications

Input Flows

cf_on_board_vehicle_data
cf_request_vehicle_data
cf_retrieved_vehicle_data

Output Flows

cf_request_on_board_vehicle_data
cf_retrieved_vehicle_data
cf_vehicle_data

Description:

Overview: This process shall be responsible for providing the communications interface and data storage facility for data that is exchanged between the commercial vehicle manager and commercial vehicle drivers in their vehicles. The process shall support the receipt of data from the vehicle consisting of that processed from input received by sensors on board the vehicle and text data used to exchange general information with the driver. Only the output to the vehicle of the data that contains the general text message shall be supported by the process. The process shall enable access to the store of received data by the manager through the manager's interface process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cf_request_vehicle_data

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'cf_on_board_vehicle_data';
(b) 'cf_retrieved_vehicle_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cf_request_on_board_vehicle_data';
(b) 'cf_vehicle_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) be responsible for the management of the data in the store of retrieved vehicle data log, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.6;
USR = 4.6.0;
USR = 4.6.1;
USR = 4.6.2;
USR = 4.6.2(a);
USR = 4.6.2(b);

Output Flow Dynamics Assumptions:

cf_request_on_board_vehicle_data = fcvm-request_on_board_vehicle_data;
cf_retrieved_vehicle_data = 1/(60*60*24);
cf_vehicle_data = fcvm-request_on_board_vehicle_data + 1/(60*60*24);

2.1.5 Provide Commercial Vehicle Driver Routing Interface

Input Flows

cf_driver_route_instructions
fcvd_request_routing_instructions

Output Flows

cf_driver_route_instructions_request
tcvd_routing_instructions

Description:

Overview: This process shall be responsible for providing the communications interface through which a commercial vehicle driver can obtain details of the vehicle route that has been provided by the commercial vehicle manager. The process shall enable the output of the route instructions in audio and/or visual form. It shall be possible for the visual form to be either hardcopy output, or in the form of a display. The process shall retain the data for a particular route internally, so that successive requests for details of the same route do not require use of the communications network.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fcvd-request_routing_instructions'.

Solicited Input Processing: This process shall receive the following data flows as a result of data being sent to another process:
(a) 'cf_driver_route_instructions'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cf_driver_route_instructions_request';
(b) 'tcvd-routing_instructions'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) on receipt of the flow identified in (a), generate the first solicited output flow shown above and await the solicited input flow produced as a response;
(c) on completion of (b) generate the second solicited output flow, taking care to present the data in a form that will be readily understood by a commercial vehicle driver..

User Service Requirements:

USR = 4.0;
USR = 4.6;
USR = 4.6.0;
USR = 4.6.1;
USR = 4.6.2;
USR = 4.6.2(a);
USR = 4.6.2(b);
USR = 4.6.2(c);

Output Flow Dynamics Assumptions:

cf_driver_route_instructions_request = fcvd-request_routing_instructions;
tcvd-routing_instructions = fcvd-request_routing_instructions;

2.1.6 Manage Driver Instruction Store

Input Flows

cf_driver_instructions
cf_driver_instructions_request
cf_driver_load_data
cf_driver_route
cf_driver_route_instructions_request

Output Flows

cf_driver_instructions
cf_driver_route_instructions
cf_driver_route_instructions_output

Description:

Overview: This process shall be responsible for managing the store of driver route instructions so that they can be loaded with data for retrieval by the commercial vehicle driver. The data for loading into the store shall be sent to the process from other processes in the Manage Commercial Vehicle Fleet Operations facility of the Manage Commercial Vehicles function. The process shall enable the data to comprise vehicle route data and vehicle load information, including the points along the route at which identified cargo is to be picked up and/or dropped off. The process shall support the retrieval of this data by the commercial vehicle driver through the driver's interface process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cf_driver_instructions_request';
(b) 'cf_driver_load_data';
(c) 'cf_driver_route';
(d) 'cf_driver_route_instructions_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'cf_driver_instructions'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cf_driver_instructions';
(b) 'cf_driver_route_instructions';
(c) 'cf_driver_route_instructions_output'.

Functional Requirements: This process shall meet the following functional requirements:
(a) run whenever any of the unsolicited input data flows shown above are received;
(b) when either the second or third unsolicited input flows is received, store the data that they contain in the store of driver instructions using the first solicited output flow, if necessary over writing data already present;
(c) when either first or fourth of the unsolicited input flows is received, retrieve the requested data from the store and send it to the requesting process in the second or third solicited output data flow;
(d) be responsible for the management of the data in the store of driver instructions, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

cf_driver_route_instructions = cf_driver_route_instructions_request;
cf_driver_route_instructions_output = cf_driver_instructions_request;
cf_driver_instructions = cf_driver_route + cf_driver_load_data;

2.2.1 Manage CV Electronic Credential and Tax Filing Interface

Input Flows

cv_driver_enrollment_payment_request
cv_driver_enrollment_request
cv_driver_route_request
cv_driver_storage_request
cv_enrollment_information
cv_enrollment_payment_confirmation
cv_route
cv_route_details
cv_static_route_data

Output Flows

cv_driver_enrollment_information
cv_driver_enrollment_payment_confirmation
cv_driver_route_data
cv_enrollment_payment_request
cv_enrollment_request
cv_route_details
cv_route_request
cv_static_route_request

Description:

Overview: This process shall be responsible for providing the commercial vehicle driver with the ability to manage the activities of a commercial vehicle. In this instance the driver is assumed to be acting in the role of a commercial vehicle manager, and is therefore probably the owner/driver of the vehicle. The process shall provide the capability for the driver to obtain commercial vehicle routes, to enroll commercial vehicles for electronic clearance at roadside check station facilities, and to process and pay for electronic credential and tax filing.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'cv_route_details', which contains data requested from and written to a data store;
(b) 'cv_enrollment_information', 'cv_enrollment_payment_confirmation', 'cv_route' and 'cv_static_route_data', which are received as the result of output to other processes.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received, generate the appropriate outputs identified below:
(c) if the input in (b) contains data to set up a route for a commercial vehicle, use the input data, plus any details of previously used routes in the local store to construct a route request and send it to either the static route selection process, or to the Provide Driver and Traveler Services function for a dynamic route, i.e. one that takes account of current and future traffic conditions;
(d) when the route requested in (c) has been provided, or the input requests that the electronic credentials be obtained and the tax details filed, send the data to the commercial vehicle administration process;
(e) repeat (d) to enable payment to be made;
(f) the process shall be responsible for the management of the data in the store of commercial vehicle route details, using the most appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.4;
USR = 4.4.1;
USR = 4.6;
USR = 4.6.1;

Output Flow Dynamics Assumptions:

cv_route_details = 1/(60*60*24)*CVO_DVR;
cv_driver_route_data = 1/(60*60*24)*CVO_DVR;
cv_driver_enrollment_information = 1/(60*60*24)*CVO_DVR;
cv_driver_enrollment_payment_confirmation = 1/(60*60*24*7)*CVO_DVR;
cv_enrollment_request = 1/(60*60*24*7)*CVO_DVR;
cv_enrollment_payment_request = 1/(60*60*24)*CVO_DVR;
cv_route_request = 1/(60*60*24)*CVO_DVR;
cv_static_route_request = 2/(60*60*24*7)*CVO_DVR;

2.2.2 Provide Vehicle Static Route

Input Flows

cv_static_route_request
map_data_for_cv_drivers

Output Flows

cv_static_route_data
map_data_for_cv_drivers

Description:

Overview: This process shall be responsible for providing a static commercial vehicle route using data provided by the commercial vehicle driver. A static route is one which is based on geographic data and therefore takes no account of current or predicted traffic conditions, incidents, etc. The process shall provide the route using its own route generation algorithms and data from its own store of digitized map information. In this instance the driver is assumed to be acting in the role of a commercial vehicle manager, and is therefore likely to be the owner/driver of the vehicle.

Data Flows: The input data flow is unsolicited and the output flow is solicited. The following data flow contains data requested from and written to a data store:
(a) 'map_data_for_cv_drivers'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the requested route and send its details in the output identified above;
(c) be responsible for the management of the data in the store of map data, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.4;
USR = 4.4.1;

Output Flow Dynamics Assumptions:

cv_static_route_data = 2/(60*60*24*7)*CVO_DVR;

2.2.3 Provide CV Driver Electronic Credential and Tax Filing Interface

Input Flows

cv_driver_credit_identity
cv_driver_enrollment_information
cv_driver_enrollment_payment_confirmation
cv_driver_route_data
cv_vehicle_data
fcvd_activity_request
fcvd_enrollment_payment_request
fcvd_enrollment_request
fcvd_other_data_input
fcvd_route_data
fcvd_route_request

Output Flows

cv_driver_enrollment_cost
cv_driver_enrollment_payment_request
cv_driver_enrollment_request
cv_driver_route_request
cv_driver_storage_request
cv_request_vehicle_data
tcvd_data_request
tcvd_enrollment_confirmation
tcvd_enrollment_payment_confirmation
tcvd_other_data_request
tcvd_route_data

Description:

Overview: This process shall be responsible for providing an interface for the commercial vehicle fleet manager. In this instance the driver is assumed to be acting in the role of a commercial vehicle manager, and is therefore likely to be the owner/driver of the vehicle. The process shall enable this interface to provide the driver with facilities for the input of data used to set up commercial vehicle routes, to pay all the necessary taxes and duties so that a commercial vehicle can be enrolled for a particular route, and to obtain a copy of the data collected by processes on-board the vehicle. The enrollment activity supported by the process shall enable a commercial vehicle to pass through the roadside checkstations along its route without stopping, unless safety checks are required. The process shall support inputs from the commercial vehicle driver in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be in hardcopy, or as a display.

Data Flows: All input data flows from the commercial vehicle driver terminator are unsolicited and all output flows are solicited. The other input flows are as a result of sending outputs to other processes.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the commercial vehicle driver terminator listed above;
(b) when the inputs are received, generate the appropriate output flows identified above and monitor for any responses.

User Service Requirements:

USR = 4.0;
USR = 4.3;
USR = 4.3.2;
USR = 4.3.2.1;
USR = 4.4;
USR = 4.4.1;
USR = 4.4.2;

Output Flow Dynamics Assumptions:

cv_driver_enrollment_cost = 1/(60*60*24*7)*CVO_DVR;
cv_driver_enrollment_request = 1/(60*60*24)*CVO_DVR;
cv_driver_enrollment_payment_request = 1/(60*60*24*7)*CVO_DVR;
cv_driver_route_request = 1/(60*60*24)*CVO_DVR;
cv_driver_storage_request = 1/(60*60*24*7)*CVO_DVR;
cv_request_vehicle_data = 1/(60*60*24);

```
tcvd-data_request = 1/(60*60*24)*CVO_DVR;
tcvd_enrollment_confirmation = 1/(60*60*24)*CVO_DVR;
tcvd_enrollment_payment_confirmation = 1/(60*60*24*7)*CVO_DVR;
tcvd-route_data = 1/(60*60*24)*CVO_DVR;
tcvd-other_data_request = 1/(60*60*24)*CVO_DVR;
```

2.2.4 Provide Commercial Vehicle Driver Communications

Input Flows

cv_on_board_vehicle_data
cv_received_vehicle_data
cv_request_vehicle_data

Output Flows

cv_received_vehicle_data
cv_request_on_board_vehicle_data
cv_vehicle_data

Description:

Overview: This process shall be responsible for providing communications between the commercial vehicle driver and the commercial vehicle. In this instance the driver is acting in the role of vehicle manager, and is therefore likely to be the owner/driver of the vehicle. The process shall support the receipt of data from the vehicle consisting of that processed from input received by sensors on board the vehicle The process shall enable access to the store of received data by the driver through the driver's interface process.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'cv_request_vehicle_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'cv_received_vehicle_data';
(b) 'cv_on_board_vehicle_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_request_on_board_vehicle_data';
)b) 'cv_vehicle_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the unsolicited input flow shown above is received, generate the first output flow, which should result in the second solicited input flow;
(c) receipt of the second solicited input flow shall generate the second solicited output flow, but this flow shall be sent with a null entry if the second input flow is not generated in (b);
(d) be responsible for the management of the data in the store of the received on-board vehicle data, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.6;
USR = 4.6.0;
USR = 4.6.1;
USR = 4.6.2;
USR = 4.6.2(a);
USR = 4.6.2(b);
USR = 4.6.2(c);

Output Flow Dynamics Assumptions:

cv_received_vehicle_data = 1/(60*60*24);
cv_request_on_board_vehicle_data = cv_request_vehicle_data;
cv_vehicle_data = 1/(60*60*24);

2.3.1 Produce Commercial Vehicle Driver Message at Roadside

Input Flows

cv_border_pull_in_output
cv_general_pull_in_output
cv_safety_pull_in_output
cv_screening_pull_in_output

Output Flows

cv_on_board_pull_in_output
tcvd_border_pull_in_output
tcvd_clearance_pull_in_output
tcvd_general_pull_in_output
tcvd_safety_pull_in_output

Description:

Overview: This process shall be responsible for the output of pull-in or pass messages to commercial vehicle drivers as they approach the commercial vehicle roadside checkstation or border crossing facilities. The process shall support the use of roadside equipment such as dynamic message signs (DMS), or simple red-green lights, flashing orange lights, etc. to provide the output. These output messages shall be received by the process from other processes responsible for roadside facilities within the Manage Commercial Vehicles function. The process shall support pull-in messages that are the result of checks on a commercial vehicle's electronic credentials, safety and border crossing data, the result of the vehicle's tag not being properly read, or the result of a general pull-in decision for all vehicles being issued by inspectors at the roadside facility. The process shall also generate a message to be sent to the vehicle so that an indication can be output directly to the driver at the same time as it appears on the roadside equipment.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the appropriate one of the three driver output flows identified above using what ever output devices are physically available;
(c) simultaneously with (b) produce the on-board pull-in output data flow and send it to the commercial vehicle..

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.4;
USR = 4.1.2;
USR = 4.1.2.2;
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.2;
USR = 4.3.1.7;

Output Flow Dynamics Assumptions:

cv_on_board_pull_in_output = cv_general_pull_in_output + cv_border_pull_in_output
+ cv_safety_pull_in_output + cv_screening_pull_in_output;
tcvd-general_pull_in_output = cv_general_pull_in_output;
tcvd-safety_pull_in_output = cv_safety_pull_in_output;
tcvd-clearance_pull_in_output = cv_screening_pull_in_output;
tcvd-border_pull_in_output = cv_border_pull_in_output;

2.3.2.1 Administer Commercial Vehicle Roadside Credentials Database

Input Flows

cv_credentials_data_request
cv_credentials_database_update
cv_credentials_information_response
cv_roadside_credentials_database

Output Flows

cv_credentials_data_output
cv_credentials_information_request
cv_roadside_credentials_database

Description:

Overview: This process shall be responsible for receiving the electronic credentials sent to the roadside checkstation facility as part of a commercial vehicle's enrollment process. The process shall store the data for use by another process and shall also enable the inspector in the roadside facility to obtain a copy of the data in the store. If the requested data is not in the store, the process shall request it from another process in the commercial vehicle administration facility.

flow is solicited and writes data to a local data store:
(a) 'cv_roadside_credentials_database'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the output identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.8;

Output Flow Dynamics Assumptions:

cv_credentials_data_output = cv_credentials_data_request;
cv_credentials_information_request = (cv_credentials_data_request)/10;
cv_roadside_credentials_database =

2.3.2.2 Process Screening Transactions

Input Flows

cv_roadside_credentials_database
cv_screening_data
cv_screening_override

Output Flows

cv_on_board_screening_record
cv_screening_decision
cv_screening_pull_in_output
cv_screening_record

Description:

Overview: This process shall be responsible for checking commercial vehicle credentials against those held in a store maintained by another process in the roadside checkstation facility. The process shall send the result of each check to the roadside inspector interface process so that an override input can be generated if required. The process shall send a request for the commercial vehicle to pull-in if the vehicle's credentials do not match those in the store, and shall also send a record of each decision to the process that maintains the commercial vehicle roadside checkstation facility log.

Data Flows: The input of screening data is unsolicited. All output flows and the other input flows are solicited, and the 'roadside_problem_credentials' flow contains data read from a store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the screening data flow listed above;
(b) when the input in (a) is received, check the data against that in the store of problem electronic credentials;
(c) if a match is found in (b) set the decision to pull-in otherwise set it to pass and send it to the inspector interface process;
(d) wait for a specified time-out period for the response to (c) and either send the amended decision in the response or the original decision as a pull-in or pass message to the commercial vehicle display interface process;
(e) encrypt the output data flow in (d) in such a way that its contents cannot be determined using any digital or analog techniques.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.4;
USR = 4.1.1.5;
USR = 4.1.1.7;
USR = 4.1.1.8;

Output Flow Dynamics Assumptions:

cv_on_board_screening_record = 20/(60*60)*ITS_CVO_VEHS;
cv_screening_pull_in_output = 20/(60*60)*ITS_CVO_VEHS;
cv_screening_decision = 20/(60*60)*ITS_CVO_VEHS;
cv_screening_record = 20/(60*60)*ITS_CVO_VEHS;

2.3.3.1 Provide Commercial Vehicle Checkstation Communications

Input Flows

cv_get_on_board_data
cv_inspection_data
cv_on_board_data

Output Flows

cv_inspection_data_output
cv_request_on_board_data
cv_roadside_collected_data

Description:

Overview: This process shall be responsible for providing an interface through which a commercial vehicles roadside checkstation facility can communicate with a passing commercial vehicle. When a request for on-board data is received from another process within the facility, the process shall issue a data request to the identified commercial vehicle. The data received by the process from the vehicle shall be stored in the store of collected data for use by the roadside inspection process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_get_on_board_data':
(b) 'cv_inspection_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'cv_on_board_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_roadside_collected_data';
(b) 'cv_inspection_data_output';
(c) 'cv_request_on_board_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) use the appropriate mechanism(s) such as RDBMS, to store data in the store of collected vehicle data.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1.4;
USR = 4.1.1.7;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.1;
USR = 4.2.2.7;
USR = 4.2.2.7(a);
USR = 4.2.2.7(b);
USR = 4.2.2.7(c);
USR = 4.2.2.7(d);
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.1;
USR = 4.3.1.3;
USR = 4.3.1.4;
USR = 4.3.1.6;
USR = 4.3.1.7;

Output Flow Dynamics Assumptions:

cv_roadside_collected_data = cv_get_on_board_data;
cv_inspection_data_output = cv_inspection_data;
cv_request_on_board_data = cv_get_on_board_data;

2.3.3.2 Provide Commercial Vehicle Inspector Handheld Terminal Interface

Input Flows

cv_inspection_results
fci_inspection_data_input
fci_start_inspection

Output Flows

cv_inspector_safety_data_input
cv_start_inspection
tci_inspection_report

Description:

Overview: This process shall be responsible for providing an interface for a hand held terminal which can be used by a commercial vehicle inspector. The process shall enable the inspector to start a commercial vehicle roadside inspection, to review the results, and to add comments to the results data. The process shall support inputs from the inspectors in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the form of the visual output to be in hardcopy, or as a display.

Data Flows: The input data flows are unsolicited with the exception of cv_inspection_results, which is solicited as are all output data flows.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) produce the inspection report in a format that is compatible with the output facilities provided by the handheld terminal, e.g. voice, liquid crystal display, etc.

User Service Requirements:

USR = 4.0;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.3;

Output Flow Dynamics Assumptions:

cv_start_inspection = fci-start_inspection;
tci-inspection_report = cv_inspection_results;
cv_inspector_safety_data_input = fci-inspection_data_input;

2.3.3.3 Administer Commercial Vehicle Roadside Safety Datadase

Input Flows

cv_roadside_safety_database_read
cv_safety_data_request
cv_safety_database_update
cv_safety_information_response

Output Flows

cv_roadside_safety_database_write
cv_safety_data_response
cv_safety_information_request

Description:

Overview: This process shall be responsible for maintaining in the commercial vehicle roadside checkstation facility a database of credentials for commercial vehicles with safety problems. This process shall store the data about these vehicles received from the commercial vehicle administration facility. It shall enable this data to be used by another process and shall also enable the inspector in the roadside facility to obtain a copy of the data in the store. If the requested data is not in the store, the process shall request it from another process in the commercial vehicle administration facility.

Data Flows: The input data flows are unsolicited. The output flows are solicited and cv_roadside_safety_database_write contains data to be written to a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, write the data to the store using the output flow identified above.

User Service Requirements:

USR = 4.0;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.4;
USR = 4.2.2.6;

Output Flow Dynamics Assumptions:

cv_roadside_safety_database_write = 1/(60*60*24);
cv_safety_data_response = cv_safety_data_request;
cv_safety_information_request = (cv_safety_data_request)/10;

2.3.3.4 Carry-out Commercial Vehicle Roadside Safety Screening

Input Flows

cv_roadside_safety_database_output
cv_safety_data
cv_safety_override

Output Flows

cv_archived_safety_data
cv_safety_decision
cv_safety_pull_in_output

Description:

Overview: This process shall be responsible for checking commercial vehicle credentials against the list of those known to have safety problems held in a store maintained by another process in the roadside checkstation facility. The process shall send the result of each check to the roadside inspector interface process so that an override input can be generated if required. The process shall send a request for the commercial vehicle to pull-in if the vehicle's credentials are in the list of those with safety problems, and shall also send a record of each decision to the process that maintains the commercial vehicle roadside checkstation facility log.

Data Flows: The input of safety data is unsolicited. All output flows and the other input flows are solicited, and the 'safety_problem_list' flow contains data read from a store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the safety data flow listed above;
(b) when the input in (a) is received, check the data against that in the store of safety problem data;
(c) if a match is found in (b) set the decision to pull-in otherwise set it to pass and send it to the inspector interface process;
(d) wait for a specified time-out period for the response to (c) and either send the amended decision in the response or the original decision as a pull-in or pass message to the commercial vehicle display interface process;
(e) encrypt the output data flow in (d) in such a way that its contents cannot be determined using any digital or analog techniques.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.1;
USR = 4.1.1.2;
USR = 4.1.1.3;
USR = 4.1.1.4;
USR = 4.1.1.6;
USR = 4.1.1.8;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.6;
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.1;
USR = 4.3.1.1(a);
USR = 4.3.1.1(b);
USR = 4.3.1.3;
USR = 4.3.1.4;
USR = 4.4;
USR = 4.4.2;

Output Flow Dynamics Assumptions:

cv_archived_safety_data = 1/(60*60*24);
cv_safety_decision = 20/(60*60)*ITS_CVO_VEHS;
cv_safety_pull_in_output = 20/(60*60)*ITS_CVO_VEHS;

2.3.3.5 Carry-out Commercial Vehicle Roadside Inspection

Input Flows

cv_inspector_safety_data_input
cv_roadside_collected_data
cv_start_inspection

Output Flows

cv_archived_inspection_data
cv_get_on_board_data
cv_inspection_data
cv_inspection_results
cv_roadside_safety_database_update
cv_update_safety_problems_list
tcvd_inspection_results

Description:

Overview: This process shall be responsible for carrying out roadside safety inspections at the request of the roadside facility inspector. The result of the inspection shall be sent by the process to the inspector, the commercial vehicle driver, the roadside checkstation facility log, and the commercial vehicle itself. The process shall enable the inspector to add comments to the result of the inspection before it is sent to the above outputs. These comments shall be received by the process in the form of data input from the inspector's hand held terminal interface.

Data Flows: The input data flow from the inspector's hand held terminal interface process is unsolicited and all output flows are solicited. The data flow 'cv_roadside_collected_data' contains data requested from a data store, and 'cv_update_problem_data' contains data to be written to a store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow requesting an inspection, as listed above;
(b) when the input is received, start the inspection process by comparing the data collected from on-board the commercial vehicle with safety criteria, generate outputs to the inspector, the roadside check facility log, the commercial vehicle and the commercial vehicle driver identified above, showing compliance or deviation;
(c) if deviatons are found send the details of the vehicle and the carrier to the safety problems list store using the flow listed above;
(d) use the appropriate mechanism(s) such as RDBMS, to retrieve data from and write data to the two stores identified above.

User Service Requirements:

USR = 4.0;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.1;
USR = 4.2.2.2;
USR = 4.2.2.3;
USR = 4.2.2.6;
USR = 4.2.2.7;
USR = 4.2.2.7(a);

Output Flow Dynamics Assumptions:

cv_archived_inspection_data = cv_start_inspection;
cv_get_on_board_data = cv_start_inspection;
cv_inspection_data = cv_start_inspection;
cv_inspection_results = cv_start_inspection;
cv_roadside_safety_database_update = CVO_FAULT_RATE*cv_start_inspection;
cv_update_safety_problems_list = CVO_FAULT_RATE*cv_start_inspection;
tcvd-inspection_results = cv_start_inspection;

2.3.4 Detect Commercial Vehicle

Input Flows

cv_electronic_clearance_data
cv_general_override
cv_manual_pull_in
fcv_vehicle_characteristics

Output Flows

cv_border_data
cv_general_decision
cv_general_pull_in_output
cv_request_electronic_clearance_data
cv_safety_data
cv_screening_data

Description:

Overview: This process shall be responsible for detecting the presence of commercial vehicles through the use of sensors that can differentiate between the different types of vehicle. The process shall use the sensors to determine the number of axles, gross vehicle weight and weight per axle data for use by inspectors at the roadside checkstation facilities. When a commercial vehicle is detected, the process shall transmit a request for its on-board tag data, which when received shall be passed to other processes within the roadside facility. If no tag data is received, or the data cannot be interpreted correctly, the process shall send a request for the vehicle to pull-in to be output by another process in the roadside checkstation facility.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the vehicle characteristics input flow shown above;
(b) when the input in (a) is received, generate the flow requesting the commercial vehicle's on-board tag data;
(c) as a result of (c) continuously monitor for receipt of the tag data flow;
(d) when the flow in (c) is received, generate the screening, safety and border data flows shown above and send them to those of the credentials, safety and border crossing processes that are present in the roadside checkstation facility;
(e) if no data is received in (c), or the data received cannot be interpreted into a set of tag data, generate the pull-in request data flow shown above and send it to the process in the roadside checkstation facility that issues pull-in requests.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.0;
USR = 4.1.1;
USR = 4.1.1.7;
USR = 4.2;
USR = 4.2.0;
USR = 4.2.1;
USR = 4.2.3;
USR = 4.2.3.2;
USR = 4.2.3.5;
USR = 4.3;
USR = 4.3.0;
USR = 4.3.1;
USR = 4.3.1.7;

Output Flow Dynamics Assumptions:

cv_border_data = fcv-vehicle_characteristics;
cv_general_decision = fcv-vehicle_characteristics;
cv_general_pull_in_output = fcv-vehicle_characteristics;
cv_request_electronic_clearance_data = fcv-vehicle_characteristics;
cv_safety_data = fcv-vehicle_characteristics;
cv_screening_data = fcv-vehicle_characteristics;

2.3.5 Provide Commercial Vehicle RoadsideOperator Interface

Input Flows

cv_border_decision
cv_credentials_data_output
cv_general_decision
cv_roadside_operator_output
cv_safety_data_response
cv_safety_decision
cv_screening_decision
fci_credentials_data_request
fci_pull_in_action
fci_request_log_report
fci_safety_data_request

Output Flows

cv_border_override
cv_credentials_data_request
cv_general_override
cv_manual_pull_in
cv_roadside_operator_data_request
cv_safety_data_request
cv_safety_override
cv_screening_override
tci_credentials_data_output
tci_output_log_report
tci_pull_in_information
tci_safety_data_output

Description:

Overview: This process shall be responsible for providing the commercial vehicle inspector interface at the roadside checkstation facility. The process shall provide an interface which enables the inspector to monitor and if necessary override the pull-in decisions made by those of the border crossing, credentials and safety data checking processes that are present in the facility. The process shall also make it possible for the inspector to issue a manual general pull-in request for all commercial vehicles to pull into the roadside checkstation facility, to have access the contents of the facility's log, and to obtain credentials or safety data on a selected combination of carrier, driver, and vehicle. The process shall support inputs from the traffic operations personnel in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be in hardcopy, or as a display.

Data Flows: All input data flows from the inspector are unsolicited and all other input and output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the inspector listed above;
(b) when the inputs in (a) are received, generate the appropriate outputs identified above;
(c) as a result of (b) monitor for the receipt of any expected responses and pass those back to the inspector through the appropriate output data flow.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.4;
USR = 4.1.1.4(a);
USR = 4.1.1.4(b);
USR = 4.1.1.4(c);
USR = 4.1.1.5;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.1;
USR = 4.2.2.2;
USR = 4.2.2.3;
USR = 4.2.2.3(a);
USR = 4.2.2.3(b);

Output Flow Dynamics Assumptions:

cv_border_override = 1/(60*60)*ITS_CVO_VEHS;
cv_credentials_data_request = fci-credentials_data_request;
cv_safety_override = 1/(60*60)*ITS_CVO_VEHS;
cv_safety_data_request = 1/(60*60)*ITS_CVO_VEHS;
cv_general_override = 1/(60*60)*ITS_CVO_VEHS;
cv_manual_pull-in = 1/(60*60)*ITS_CVO_VEHS;
cv_safety_information_request = fci-safety_data_request;
cv_roadside_operator_data_request = 1/(60*60)*CVO_INSP;
cv_screening_override = 1/(60*60)*ITS_CVO_VEHS;
tci-credentials_data_output = fci-credentials_data_request;
tci-pull-in_information = 4/(60*60)*CVO_INSP;
tci-output_log_report = 4/(60*60)*CVO_INSP;
tci-safety_data_output = fci-safety_data_request;

2.3.6 Provide Commercial Vehicle Reports

Input Flows

cv_archived_inspection_data
cv_archived_safety_data
cv_border_record
cv_roadside_facility_log
cv_roadside_operator_data_request
cv_screening_record

Output Flows

cv_border_daily_log
cv_roadside_daily_log
cv_roadside_facility_log
cv_roadside_operator_output

Description:

Overview: This process shall be responsible for collecting data from those of the border crossing, credential and safety checking processes that are present in a commercial vehicle roadside checkstation facility. The data shall be stored by the process in a roadside facility log, to which the roadside inspector interface process shall have access. Once a day the process shall make a copy of the roadside facility log and send it to the commercial vehicle administration facility for further processing.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'cv_roadside_facility_log', which contains data requested from or written to a data store;
(b) 'cv_roadside_daily_log', which is sent daily without being generated by input from other processes.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when any inputs which are not a request for the contents of the log are received, load the data that they contain into the log;
(c) when the input requests output of the contents of the log, retrieve the data and send it to the source of the request;
(d) regardless of any other similar requests, daily read the data from the log and send it to the Administer Commercial Vehicles facility;
(e) be responsible for the management of the data in the roadside facility log, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.0;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.4;
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.3;
USR = 4.4;
USR = 4.4.2;
USR = 4.4.2(b);

Output Flow Dynamics Assumptions:

cv_roadside_facility_log = 1/HOUR*CVO_INSP;
cv_roadside_operator_output = 1/HOUR*CVO_INSP;
cv_roadside_daily_log = 1/DAY;
cv_border_daily_log = 1/DAY;

2.3.7 Produce Commercial Vehicle Driver Message on Vehicle

Input Flows

cv_on_board_pull_in_output

Output Flows

tcvd_on_board_pull_in_output

Description:

Overview: This process shall be responsible for the output of the pull-in or pass messages to commercial vehicle drivers directly in their vehicles as they approach a commercial vehicle roadside checkstation facility. These messages shall be generated by other processes within the facility that are responsible for checking the commercial vehicle's credentials (including those for border crossing) and safety, or may be the result of the vehicle's tag not being properly read, or may be the result of a general pull-in decision for all vehicles being issued by inspectors at the roadside checkstation facility.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the output flow identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.4;
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.2;
USR = 4.3.1.7;

Output Flow Dynamics Assumptions:

tcvd-on_board_pull_in_output = cv_on_board_pull_in_output;

2.3.8 Provide Commercial Vehicle Border Screening

Input Flows

cv_border_data
cv_border_database_update
cv_border_override
cv_roadside_border_database

Output Flows

cv_border_decision
cv_border_pull_in_output
cv_border_record
cv_on_board_border_record
cv_roadside_border_database

Description:

Overview: This process shall be responsible for checking a commercial vehicle and its cargo through a border crossing point. The checks carried out by the process shall comprise a comparison of the trip identity already provided by the commercial vehicle administration processes, and held in a local data store. A check shall also be made by the process to see if the lock tag attached to the vehicle's cargo has been changed. If either of these two checks produce negative results then the process shall request the vehicle to pull-in, otherwise the vehicle shall be allowed to pass. The process shall send its decision to the process that provides the roadside inspectors' interface, to enable an override to be applied if required. The decision of the process (with the override if it is applied) shall be sent to the message output process and be written back to the vehicle's on-board tag.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_border_data';
(b) 'cv_border_database_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores, or data sent to another process:
(a) 'cv_roadside_border_database';
(b) 'cv_border_override'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_border_decision';
(b) 'cv_border_pull_in_output';
(c) 'cv_on_board_border_record';
(d) 'cv_roadside_border_database'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for the unsolicited inputs flow listed above;
(b) when the first unsolicited input flow is received, check the trip number against the data that is already contained in the local data store and that the lock tag data shows that it has not been touched;
(c) if a match is found in (b) and the lock tag is present and has not been tampered, then generate a pass decision, otherwise generate a pull-in decision;
(d) generate the first solicited output data flow shown above with the pull-in/pass decision that results from (c);
(e) as a result of (d) continuously monitor for a specified period of time for receipt of the second solicited input data flow listed above;
(f) if no data is received in (e), maintain the current decision, otherwise change the decision to that received in the flow;
(g) as a result of (f) output the pull-in/pass decision in the second solicited output data flow listed above, and also generate the fourth solicited output data flow listed above;
(h) encrypt the fourth solicited output data flow in such a way that its contents cannot be determined using any digital or analog techniques;
(i) when the second unsolicited input flow is received in (a), load the data that it contains into the local data store;
(j) use the appropriate mechanism(s) such as RDBMS, to retrieve data from and write data to the other stores identified above.

User Service Requirements:

USR = 4.0;

USR = 4.1;
USR = 4.1.2;
USR = 4.1.2.2;
USR = 4.4.3;
USR = 4.4.3.1;
USR = 4.4.3.1(a);
USR = 4.4.3.1(b);
USR = 4.4.3.1(c);
USR = 4.4.3.2;
USR = 4.4.3.2(a);
USR = 4.4.3.2(b);
USR = 4.4.3.2(c);
USR = 4.4.3.2(d);
USR = 4.4.3.2(e);
USR = 4.4.3.2(f);
USR = 4.4.3.2(g);
USR = 4.4.3.2(h);

Output Flow Dynamics Assumptions:

cv_border_decision = cv_border_data;
cv_border_pull_in_output = cv_border_data;
cv_on_board_border_record = cv_border_data;
cv_roadside_border_database = cv_border_database_update;
cv_border_record = cv_border_data;

2.4.1 Communicate Commercial Vehicle On-board Data to Roadside

Input Flows

cv_inspection_data_output
cv_on_board_data_current_copy
cv_request_on_board_data

Output Flows

cv_inspection_data_update
cv_on_board_data
cv_on_board_data_needed

Description:

Overview: This process shall be responsible for providing the commercial vehicle end of the communications link between itself and a commercial vehicle roadside checkstation facility. The process shall enable an inspector at the facility or elsewhere to have access to the data accumulated on-board the vehicle for use in a vehicle inspection. It shall also enable the inspector to send back data about the result of the inspection for storage on-board the vehicle.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_inspection_data_output';
(b) 'cv_request_on_board_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'cv_on_board_data_current_copy'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_inspection_data_update';
(b) 'cv_on_board_data';
(c) 'cv_on_board_data_needed'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the second unsolicited data flow is received generate the third solicited output data flow and await receipt of the solicited input data flow shown above;
(c) when the data flow in (b) is received, generate the second solicited data flow shown above;
(d) when the first unsolicited data flow in (a) is received, generate the first solicited output data flow shown above.

User Service Requirements:

USR = 4.0;
USR = 4.2;
USR = 4.2.3;
USR = 4.2.3.3;
USR = 4.2.3.7;
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.6;
USR = 4.3.2;
USR = 4.3.2.3;

Output Flow Dynamics Assumptions:

cv_inspection_data_update = cv_inspection_data_output;
cv_on_board_data = cv_request_on_board_data;
cv_on_board_data_needed = cv_request_on_board_data;

2.4.2 Collect On-board Commercial Vehicle Sensor Data

Input Flows

fcv_brake_condition
fcv_cargo_safety_status
fcv_distance_travelled
fcv_driver_safety_status
fcv_driver_status
fcv_vehicle_safety_status
fcv_weight

Output Flows

cv_on_board_stored_sensor_data

Description:

Overview: This process shall be responsible for continuously monitoring the conditions on-board a commercial vehicle. These inputs shall be processed by sensors, and if required converted from analog into a digital form. The process shall load all collected into an on-board vehicle data store for use by other processes in the vehicle.

Data Flows: All input data flows are unsolicited and the output flow is solicited containing data to be written to a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously generate the output of data to the data store listed above, using data scanned from the inputs also listed above;
(b) complete a full scan of all inputs and generate the output in a period of time that is consistent with the safe operation of vehicle control systems with human interfaces, regardless of the number of inputs to be scanned;
(c) be capable of accepting input data in a variety of formats, both digital and analog.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.6;
USR = 4.1.2;
USR = 4.1.2.1;
USR = 4.2;
USR = 4.2.0;
USR = 4.2.3;
USR = 4.2.3.1;
USR = 4.2.3.2;
USR = 4.2.3.3;
USR = 4.2.3.4;
USR = 4.2.3.5;
USR = 4.2.3.6;
USR = 4.3;
USR = 4.3.1;
USR = 4.3.1.2;
USR = 4.3.2;
USR = 4.3.2.1;

Output Flow Dynamics Assumptions:

cv_on_board_stored_sensor_data = 1/(60);

2.4.3 Analyze Commercial Vehicle On-board Data

Input Flows

cv_driver_data_input
cv_on_board_stored_sensor_data

Output Flows

cv_critical_safety_problem
cv_driver_data_output
cv_on_board_data_update

Description:

Overview: This process shall be responsible for analyzing the data collected on-board a commercial vehicle, and sending it to another process for loading into a store on-board the vehicle. If the analysis of the data carried out by the process shows that there is a critical safety problem, the process shall send data to the driver's interface process for output to the driver. The process shall also accept input of data from the commercial vehicle driver via the interface process and load it into the same store.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'cv_on_board_stored_sensor_data', which contains data requested from a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow containing driver input shown above;
(b) when the input in (a) is received, generate the solicited output flow that send the data it contains to the process that manages the store of on-board vehicle data;
(c) periodically read the contents of the data store containing commercial vehicle sensor data, analyze the contents and again generate the solicited output flow that send the data it contains to the process that manages the store of on-board vehicle data;
(d) if during the analysis defined in (c) a critical safety problem is found, generate the output flow that highlights the problem to the commercial vehicle driver interface process;
(e) read the data from the store of commercial vehicle sensor data, using the most appropriate mechanism(s) such as RDBMS.

User Service Requirements:

USR = 4.2.3.1;
USR = 4.2.3.2;
USR = 4.2.3.4;
USR = 4.2.3.5;
USR = 4.3.2.1;
USR = 4.3.2.1(a);
USR = 4.3.2.1(b);
USR = 4.3.2.1(c);
USR = 4.3.2.1(d);
USR = 4.3.2.1(e);
USR = 4.3.2.2;
USR = 4.4.3.2;

Output Flow Dynamics Assumptions:

cv_on_board_data_update = 12/(60*60);
cv_critical_safety_problem = 1/(60*60*24*7);
cv_driver_data_output = 1/(60*60*24);

2.4.4 Provide Commercial Vehicle Driver Interface

Input Flows

cv_critical_safety_problem
cv_driver_data_output
cv_general_input_message
fcvd_driver_data_input
fcvd_driver_general_message
fcvd_driver_input_type

Output Flows

cv_driver_data_input
cv_general_output_message
cv_output_on_board_vehicle_data
tcvd_critical_safety_problem
tcvd_data_input_request
tcvd_output_data
tcvd_type_input_request

Description:

Overview: This process shall be responsible for providing the interface between the commercial vehicle driver and processes on-board the commercial vehicle. The process shall provide interfaces to the processes responsible for collecting, analyzing and storing data about the vehicle, its cargo, the driver, etc., and for the exchange of data with the commercial vehicle manager. The process shall support inputs from the driver in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be in hardcopy, or as a display.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs from the driver are received, generate the appropriate outputs to the other processes;
(c) when inputs from other processes are received, generate the appropriate outputs identified above to the driver.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.6;
USR = 4.1.1.6(a);
USR = 4.1.1.6(b);
USR = 4.1.1.6(c);
USR = 4.1.1.6(d);
USR = 4.1.1.6(e);
USR = 4.0;
USR = 4.3;
USR = 4.3.2;
USR = 4.3.2.1;
USR = 4.4;
USR = 4.4.1;
USR = 4.4.2;
USR = 4.4.3;
USR = 4.4.3.1;

Output Flow Dynamics Assumptions:

cv_general_output_message = fcvd-driver_general_message;
cv_output_on_board_vehicle_data = 1/(60*60*24*7);
cv_driver_data_input = 1/(60*60*24);
tcvd-critical_safety_problem = 1/(60*60*24*7); USR = 4.4.3.1(a);
tcvd-data_input_request = 5/(60*60); USR = 4.4.3.1(b);
tcvd-output_data = 3/(60*60); USR = 4.4.3.1(c);
tcvd-type_input_request = 2/(60*60); USR = 4.4.3.2;
USR = 4.4.3.2(a);

2.4.5 Communicate Commercial Vehicle On-board Data to Vehicle Manager

Input Flows

cf_request_on_board_vehicle_data
cv_general_output_message
cv_on_board_data_output
cv_output_on_board_vehicle_data
cv_request_on_board_vehicle_data
vehicle_location_for_cv

Output Flows

cf_on_board_vehicle_data
cv_general_input_message
cv_on_board_data_required
cv_on_board_vehicle_data

Description:

Overview: This process shall be responsible for providing the communications interface through which the commercial vehicle manager (or commercial vehicle driver acting in the role of the manager) can access the data stored on-board a commercial vehicle. The process shall also support the exchange of unformatted messages between the commercial vehicle manager and driver, and the ability of the driver to send the on-board data to the manager as an unsolicited data flow.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows containing requests from the commercial vehicle manager or driver listed above for output of the on-board vehicle data;
(b) when these inputs are received, generate the output data flow to the store management process that requests a copy of the data currently loaded into the store;
(c) monitor for receipt of the data flow containing the current store contents in response to the data flow sent in (b);
(d) when the data flow in (c) is received, generate the data flow to send the data to commercial vehicle manager or driver who requested the data;
(e) if the data flow from the commercial vehicle manager contains a general output message, generate the
data flow that sends this data for output by the commercial vehicle driver interface process;
(f) if the input data flow with the general message is received from the commercial vehicle driver interface process, generate the output to the commercial vehicle manager but without the on-board vehicle data, only include the general message;
(g) if the input data flow with the request for output of the on-board data is received from the commercial vehicle driver interface process, generate the output flow in (b) and follow through with step (c) above;
(h) as a result of (g) send the on-board data to the commercial vehicle manager.

User Service Requirements:

USR = 4.2.3.1;
USR = 4.2.3.2;
USR = 4.2.3.4;
USR = 4.2.3.5;
USR = 4.3.2.1;
USR = 4.3.2.2;
USR = 4.4.3.2;

Output Flow Dynamics Assumptions:

cv_on_board_data_required = cf_request_on_board_vehicle_data+cv_request_on_board_vehicle_data
+cv_output_on_board_vehicle_data;

2.4.6 Provide Commercial Vehicle On-board Data Store Interface

Input Flows

cv_inspection_data_update
cv_on_board_data_needed
cv_on_board_data_required
cv_on_board_data_update
cv_on_board_stored_data
cv_provide_credentials_data_for_inspections

Output Flows

cv_on_board_data_current_copy
cv_on_board_data_output
cv_on_board_stored_data

Description:

Overview: This process shall be responsible for providing the interface through which data can be written to and read from the store of data that is held on-board a commercial vehicle. The data shall be provided by and on request from, other processes within the Manage Commercial Vehicles function that are on-board the vehicle.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'cv_on_board_stored_data', which contains data requested from or written to a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows containing updates to the data in the store or request the output of the current contents of the store;
(b) when any of the update inputs are received, write the new data into the store;
(c) when any of the requests for data output are received, read the current contents of the store and generate the appropriate output flow to send the data to the requesting process;
cv_on_board_vehicle_data = cv_request_on_board_vehicle_data;
(d) when the data flow with the credentials details is received, load the data into the store
cv_general_input_message = cf_request_on_board_vehicle_data;
overwriting any similar data that store already contains;
cf_on_board_vehicle_data = cf_request_on_board_vehicle_data+cv_output_on_board_vehicle_data;
(e) be responsible for the management of the data in the store of commercial vehicle data using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 4.2.3.1;
USR = 4.2.3.2;
USR = 4.2.3.4;
USR = 4.2.3.5;
USR = 4.3.2.1;
USR = 4.3.2.2;
USR = 4.4.3.2;

Output Flow Dynamics Assumptions:

cv_on_board_data_current_copy = cv_on_board_data_needed;
cv_on_board_data_output = cv_on_board_data_required;
cv_on_board_stored_data = cv_on_board_data_update+cv_inspection_data_update
+cv_provide_credentials_data_for_inspections;

2.5.1 Manage Commercial Vehicle Trips and Clearances

Input Flows

cf_enrollment_payment_request
cf_enrollment_request
cf_tax_audit_data
cv_check_credentials_response
cv_confirmed_enrollment
cv_enrollment_payment_request
cv_enrollment_request
cv_remote_enrollment_confirmation
cv_request_enrollment_data
cv_tax_and_credential_fees
cv_update_new_credentials_response

Output Flows

cf_enrollment_information
cf_enrollment_payment_confirmation
cv_check_credentials_request
cv_enrollment_information
cv_enrollment_list
cv_enrollment_payment_confirmation
cv_provide_enrollment_data
cv_remote_enrollment_request
cv_request_permits_and_duties_update
cv_update_new_credentials_request

Description:

Overview: This process shall be responsible for the advance acquisition of electronic credentials and tax filing for commercial vehicles. The process will support the payment of the necessary taxes and duties that will enable a vehicle to be cleared through the credentials checks at the roadside checkstation facilities along its route, including those at border crossings. For this activity the process uses information about the vehicle's route provided by the commercial vehicle manager, or by the driver acting in that role when the vehicle is owned and operated by the driver. The actual payment activity and the subsequent notification of the roadside facilities along the route is carried out by other processes. Where the roadside facilities are outside the area served by the local ITS functions, the process requests that the necessary vehicle data is passed to the similar processes serving the appropriate areas.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cf_enrollment_payment_request';
(b) 'cf_enrollment_request';
(c) 'cv_enrollment_payment_request';
(d) 'cv_enrollment_request';
(e) 'cv_request_enrollment_data';
(f) 'cf_tax_audit_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and queries to local data stores:
(a) 'cv_check_credentials_response';
(b) 'cv_confirmed_enrollment';
(c) 'cv_remote_enrollment_confirmation';
(d) 'cv_tax_and_credential_fees';
(e) 'cv_update_new_credentials_response'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cf_enrollment_information';
(b) 'cf_enrollment_payment_confirmation';
(c) 'cv_check_credentials_request';
(d) 'cv_enrollment_information';
(e) 'cv_enrollment_list';
(f) 'cv_enrollment_payment_confirmation;
(g) 'cv_provide_enrollment_data';
(h) 'cv_remote_enrollment_request';
(i) 'cv_request_permits_and_duties_update';
(j) 'cv_update_new_credentials_request'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the inputs containing enrollment requests are received, either from a local process or from the process that communicates with other administer commercial vehicles facilities, send the output data flow to check that the enrollment can be completed, using the supplied credentials;
(c) when the response to (b) is received, generate the enrollment information data flow and sent it back to the process from which the request was sent;
(d) when inputs containing enrollment payment information are received, either from a local process or from the process that communicates with other administer commercial vehicles facilities, generate the enrollment list data flow including in it the cost of the necessary payments and duties that are in the store of tax and credential fees, and send it to the process that obtains payment;
(e) monitor for the response to (d), and when received, generate the update enrollment data flow;
(f) when the response to (e) is received, generate the enrollment payment confirmation data flow and send it to the process from which the request in (d) was received;
(g) periodically send a request to the process that interfaces to the government administrators to request an update to the tax and credentials data;
(h) use the appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.6;
USR = 4.6.1;

Output Flow Dynamics Assumptions:

cf_enrollment_information = 1/DAY*CVO_VEHS;
cf_enrollment_payment_confirmation = 4/WEEK*CVO_VEHS;
cv_check_credentials_request = 1/DAY*CVO_VEHS+1/WEEK*CVO_DVR;
cv_enrollment_information = 1/WEEK*CVO_DVR;
cv_enrollment_payment_confirmation = 4/WEEK*CVO_DVR;
cv_enrollment_list = 4/WEEK*ITS_CVO_VEHS+1/DAY*CVO_DVR;
cv_remote_enrollment_request = 1/WEEK*CVO_VEHS;
cv_provide_enrollment_data = cv_remote_enrollment_request;
cv_request_permits_and_duties_update = 4/WEEK*CVO_VEHS;
cv_update_new_credentials_request = 4/WEEK*CVO_VEHS+4/WEEK*CVO_DVR;

2.5.2 Obtain Electronic Credential and Tax Filing Payment

Input Flows

cv_enrollment_list
cv_tax_and_credential_fees
financial_response

Output Flows

cv_confirmed_enrollment
financial_request

Description:

Overview: This process shall be responsible for making payment for electronic credential and tax filing. The data on which the payment is based shall be that for a commercial vehicle's route as provided by the commercial vehicle manager or the commercial vehicle driver who is also the owner of the vehicle. The actual payment activity will be carried out by another process in the Provide Electronic Payment Services function.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_enrollment_list'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'authorization_code';
(b) 'cf_credit_identity':
(c) 'cv_tax_and_credential_fees'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_confirmed_enrollment';
(b) 'financial_request'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) use the appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.6;
USR = 4.6.1;

Output Flow Dynamics Assumptions:

cv_confirmed_enrollment = 4/(60*60*24*7)*ITS_CVO_VEHS+1/(60*60*24)*CVO_DVR;
financial_request = 4/(60*60*24*7)*ITS_CVO_VEHS+1/(60*60*24)*CVO_DVR;

2.5.3 Update Permits and Duties Store

Input Flows

cv_request_permits_and_duties_update
cv_roadside_facility_locations
fga_carrier_safety_ratings
fga_roadside_facility_locations
fga_tax_and_credential_fees

Output Flows

cv_roadside_facility_location
cv_roadside_facility_locations
cv_safety_history_update
cv_tax_and_credential_fees
tga_request_fees_updates

Description:

Overview: This process shall be responsible for receiving data from Government Administrators. This data comprises updates to the list of electronic credentials and tax filing required for a commercial vehicle to pass each roadside checkstation facility, plus carrier safety ratings for use in roadside safety inspections. These updates are both loaded into a store used by other process in the commercial vehicle administration facility.

Data Flows: The input data flow requesting an update to the data is unsolicited and all output flows are solicited. The following data flows contain data to be written to a data store:
(a) 'cv_tax_and_credential_fees';
(b) 'cv_safety_history_update'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above which requests an update of the data from the Government Administrators;
(b) when the input is received, generate the output identified above to the Government Administrators;
(c) when data is received from the Government Administrators load it into the data stores using the two flows identified above;
(d) use the appropriate mechanism(s) such as RDBMS, to write data to the two stores identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.2;
USR = 4.4.0;
USR = 4.4.1;
USR = 4.4.1(a);
USR = 4.4.1(b);
USR = 4.4.3;
USR = 4.4.3.2;

Output Flow Dynamics Assumptions:

cv_roadside_facility_locations = fga-roadside_facility_locations;
cv_tax_and_credential_fees = cv_request_permits_and_duties_update;
cv_safety_history_update = fga-carrier_safety_ratings;
tga-request_fees_updates = cv_request_permits_and_duties_update;
cv_roadside_facility_location = fga-roadside_facility_locations;

2.5.4 Communicate with Other Commercial Vehicle Administration System

Input Flows

cv_commit_remote_enrollment
cv_provide_enrollment_data
cv_remote_enrollment_request
focvas_commit_local_enrollment
focvas_data_table
focvas_enrollment_confirmation
focvas_enrollment_request
focvas_provide_data

Output Flows

cv_commit_local_enrollment
cv_remote_enrollment_confirmation
cv_request_enrollment_data
tocvas_commit_remote_enrollment
tocvas_data_table
tocvas_enrollment_confirmation
tocvas_enrollment_request
tocvas_provide_data

Description:

Overview: This process shall be responsible for communicating with commercial vehicle administration facilities in ITS functions that serve areas outside that which is served by the local function. The communications supported by the process shall enable the local function to enroll commercial vehicles in other areas, and for those other areas to enroll their commercial vehicles in the local area. The process shall thus support the coordination and the determination of electronic credentials and tax filing across geographic and jurisdictional boundaries.

Data Flows: With the exception of the input data flows listed below which are unsolicited, all other input and output flows are solicited as a result of input of either of these flows, with the exception of the 'cv_tax_and_credential_fees' data flow which contains data requested from a local data store.
(a) 'cv_remote_enrollment_request';
(b) 'focvas-enrollment_request'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the two unsolicited input flows listed above;
(b) when the local input flow is received, generate the output identified above requesting data from the remote system;
(c) when the data requested in (b) arrives from the remote system send it to the requesting process;
(d) when the input requesting data from the remote system is received, collect the data from the local store and process, and send it back to the remote system;
(e) use the appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.3;
USR = 4.6;
USR = 4.6.1;

Output Flow Dynamics Assumptions:

cv_commit_local_enrollment = focvas-commit_local_enrollment;
cv_remote_enrollment_confirmation = focvas-enrollment_request;
cv_request_enrollment_data = focvas-enrollment_request;
tocvas-commit_remote_enrollment = cv_commit_remote_enrollment;
tocvas-data_table = focvas-enrollment_request;
tocvas-enrollment_confirmation = focvas-enrollment_request;
tocvas-enrollment_request = focvas-enrollment_request;
tocvas-provide_data = focvas-enrollment_request;

2.5.5 Manage Commercial Vehicle Credentials and Enrollment

Input Flows

cv_check_credentials_request
cv_commit_local_enrollment
cv_roadside_facility_location
cv_safety_history
cv_update_new_credentials_request
fcvoir_request_for_information
fea_cv_enforcement_agency_response

Output Flows

cv_check_credentials_response
cv_commit_remote_enrollment
cv_credentials_enrollment_data
cv_database
cv_update_new_credentials_response
tcvoir_carrier_or_vehicle_information
tea_cv_request_for_information

Description:

Overview: This process shall be responsible for enabling commercial vehicle managers and drivers (who are owners) to enroll the electronic credentials for their vehicles. This enrollment shall be achieved by loading the credentials data into a data store from which such data shall be downloaded to the commercial vehicle roadside checkstation and border crossing facilities by another process. When the roadside facility is located in the area not served by the local Manage Commercial Vehicles function, the process sends the data to another process that is responsible for communicating with a similar function in other geographic and/or jurisdictional areas. The process shall also be able to accept commercial vehicle enrollment requests from similar functions in other areas, query enforcement agency databases for outstanding prosecutions, and shall be able to respond to requests for information from authorized entities, such as insurance underwriters.

Data Flows: All input data flow containing the enrollment requests from commercial vehicle managers and drivers are unsolicited. All output flows are solicited. The following data flows write data to, or receive data from data stores:
(a) 'cv_database', write only;
(b) 'cv_safety_history', read only.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above containing the enrollment requests from either commercial vehicle managers or drivers, or from a remote commercial vehicle administration system, or from an external entity;
(b) when the inputs from the commercial vehicle managers or drivers are received, enroll the data by updating the database of commercial vehicle credential data identified above for the roadside checkstation facilities that are on the vehicle's route;
(c) as part of (b), send requests for enrollment to any roadside facilities that are on the route but not in the area served by the function to the process that interfaces to other commercial vehicle administration systems using the commit data flow;
(d) before carrying out any enrollments, send the output to the enforcement agency to check that there are no outstanding prosecutions for the carrier/driver/vehicle combination that might prevent the vehicle being cleared to pass a roadside checkstation facility;
(e) when input is received from the information reqesters, obtain the required data from the store of safety data and send it back to the requester;
(f) be responsible for the management of the data in the database of commercial vehicle credential data, using the appropriate mechanism(s) such as RDBMS, for storing the data.
(g) also use the appropriate mechanism(s) such as RDBMS, to retrieve data from the other store
(h) identified above.
(h) send commercial vehicle archive data to be stored for further processing by an archive manager.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.8;
USR = 4.2;
USR = 4.2.2;

USR = 4.2.2.1;
USR = 4.2.2.4;
USR = 4.4.0;
USR = 4.4.1;
USR = 4.4.1(a);
USR = 4.4.1(b);
USR = 4.4.2;

Output Flow Dynamics Assumptions:

cv_check_credentials_response = cv_check_credentials_request;
cv_commit_remote_enrollment = (7/10)*cv_update_new_credentials_request;
cv_database = cv_update_new_credentials_request;
cv_update_new_credentials_response = cv_update_new_credentials_request;
tea-cv_request_for_information = cv_update_new_credentials_request;
tcvoir-carrier_or_vehicle_information = fcvoir-request_for_information;
cv_credentials_enrollment_data = cv_update_new_credentials_request;

2.5.6 Output Commercial Vehicle Enrollment Data to Roadside Facilities

Input Flows

cv_credentials_information_request
cv_database
cv_facility_log
cv_safety_history_read
cv_safety_information_request

Output Flows

cv_border_database_update
cv_credentials_database_update
cv_credentials_information_response
cv_safety_database_update
cv_safety_information_response

Description:

Overview: This process shall be responsible for providing credentials, safety and border crossing data to commercial vehicle roadside checkstation and border crossing facilities. This data shall be output by the process periodically (e.g. daily) from an interrogation of the stores of safety history and credentials, and sent to the roadside facilities served by the local Manage Commercial Vehicles function. The process shall also provide selected credentials and safety data on request from the commercial vehicle inspectors at particular roadside checkstation facilities.

Data Flows: There are no unsolicited input data flows and all output flows are solicited. Two input flows are generated as a result data queries to the following data stores:
(a) 'cv_facility_log';
(b) 'cv_safety_history'.

Functional Requirements: This process shall meet the following functional requirements:
(a) periodically, e.g. daily, request data from the two data stores identified above;
(b) when the data has been received, generate the outputs identified above;
(c) when any of the requesting data flows are received, retrieve the required data using the same flows as used in (a) above;
(d) when the data has been received, generate the appropriate response flows identified above;
(e) use the appropriate mechanism(s) such as RDBMS, to retrieve data from the data stores identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.8;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.1;
USR = 4.2.2.5;
USR = 4.2.2.5(a);
USR = 4.2.2.5(b);
USR = 4.2.2.5(c);
USR = 4.2.2.5(d);
USR = 4.2.2.4;

Output Flow Dynamics Assumptions:

cv_border_database_update = 1/DAY;
cv_credentials_database_update = 1/DAY;
cv_credentials_information_response = cv_credentials_information_request;
cv_safety_database_update = 1/DAY;
cv_safety_information_response = cv_safety_information_request;

2.5.7 Process Commercial Vehicle Violations

Input Flows

cv_facility_log

Output Flows

cv_violation_data

Description:

Overview: This process shall be responsible for sending details of commercial vehicle carriers and drivers that require prosecution to a process in the Manage Emergency Services function. The receiving process in that function will be responsible for sending the data to the appropriate law enforcement agency. This process shall obtain the data by periodically (e.g. daily) scanning the data in the log obtained from the commercial vehicle roadside checkstation facilities.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'cv_facility_log'.

Unsolicited Output Processing: This process shall provide the following output flows regardless of any inputs that are received:
(a) 'cv_violation_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) run periodically (e.g. daily) and read the data from the store of log data obtained from commercial vehicle roadside checking and border facilities;
(b) where any safety or credential problems have occurred that could make the driver and/or carrier liable for prosecution, send the data to the law enforcement interface process in the Manage Emergency Services function;
(c) use the appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.4;
USR = 4.4.3;
USR = 4.4.3.2;
USR = 4.2.0;
USR = 4.2.1;
USR = 4.2.2;

Output Flow Dynamics Assumptions:

cv_violation_data = 1/(60*60*24);

2.5.8 Process Data Received from Roadside Facilities

Input Flows

cf_request_activity_report
cv_border_daily_log
cv_facility_log
cv_roadside_daily_log
cv_update_safety_problems_list

Output Flows

cf_periodic_activity_report
cf_roadside_activity_report
cv_daily_logs
cv_facility_log
cv_safety_history_write
tga_quarterly_reports

Description:

Overview: This process shall be responsible for the examination of the daily logs received periodically from the commercial vehicle checkstation and border crossing facilities. It shall also be responsible for the receipt in real time of data about commercial vehicles that have failed their safety inspections. The examination of the received data shall lead the process to update the local stores containing the facility logs and vehicle safety history. This process shall also issue quarterly reports for use by government administrators, send details of the activity at the roadside facility to the Plan System Deployment function. It shall also provide responses to requests from the commercial vehicle manager for reports of fleet activity through roadside facilities, either on-demand or as periodic summaries..

Data Flows: All input data flow containing the roadside and border daily logs, plus the enrollment requests from commercial vehicle fleet managers and drivers are unsolicited. All output flows are solicited. The following data flows write data to, or receive data from data stores:
(a) 'cv_facility_log', read and write;
(b) 'cv_safety_history_write', write only.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above containing the facility logs, details of commercial vehicles that have failed their safety inspections, and requests from the commercial vehicle manager for activity reports;
(b) when the input of facility logs are received, update the roadside log data store identified above, and if necessary the store of safety history data;
(c) when processing the facility log, remove all traces of vehicle, carrier and driver identity and produce a log of activity which is sent to the Plan System Deployment function using the output flow listed above;
(d) when an activity report request is received, generate the appropriate report data and send it to the requesting commercial vehicle manager;
(e) if a periodic report request is received, generate the first report as in (d) and generate the subsequent reports at the requested intervals;
(f) be responsible for the management of the data in the store of the facility log, using the appropriate mechanism(s) such as RDBMS, for storing the data;
(g) also use the appropriate mechanism(s) such as RDBMS, to write data to the other store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.1;
USR = 4.1.1.8;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.1;
USR = 4.2.2.4;
USR = 4.2.2.5;
USR = 4.4;
USR = 4.4.2;
USR = 4.4.2(a);
USR = 4.4.2(c);
USR = 4.4.2(d);

Output Flow Dynamics Assumptions:

cf_roadside_activity_report = cf_request_activity_report;
cf_periodic_activity_report = CVO_MAN/MONTH;
cv_operational_data = 1/DAY;
cv_facility_log = 1/DAY;
cv_safety_history_write = 1/DAY;
tga-quarterly_reports = 1/(90*DAY);
cv_daily_logs = cv_roadside_daily_log + cv_border_daily_log;

2.5.9 Manage Commercial Vehicle Archive Data

Input Flows

cv_archive_request
cv_archive_status
cv_credentials_enrollment_data
cv_daily_logs
cv_data_archive

Output Flows

cv_archive_data
cv_data_archive

Description:

Overview: This process shall be responsible for processing request for archive data of commericial vehicle operations. This process shall receive operational data from the roadside check systems and administration and credentials data. This process shall receive and respond to requests from the Manage Archived Data process for either a catalog of the data contained with the commercial vehicle data stores or for the data itself. Additionally this process shall be able to produce sample products of the data available. As data is received into this process quality control metrics shall be assigned. The appropriate meta-data shall be generated and store along with the data. A catalog of the data shall be maintained to allow requesters to know what data is available from the archive store. The process shall run when a request for data is received from an external source, or when fresh data is received.

All inputs to this process are unsolicited, and all outputs are solicited, except that the 'cv_archive_status' is a solicited input.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited data inputs shown above is received, the process shall store them in the data store along with meta data (data attributes about the data), and update the catalog;
(c) when the request for cv archive data is received, the process shall immediately generate the solicited output shown above from the data store;
(d) the process should then receive the cv archive status solicited input;
(e) data shall only be sent to the source from which the data request originated;
(f) before output, the process shall put the data into a format that is easily read and interpreted by external processes and can also be read by travelers and toll users with the minimum of further processing.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.6;
USR = 7.1.3.1.6(a);
USR = 7.1.3.1.6(b);
USR = 7.1.3.1.6(c);
USR = 7.1.3.1.6(d);
USR = 7.1.3.1.6(e);
USR = 7.1.3.1.6(f);

Output Flow Dynamics Assumptions:

cv_archive_data = cv_archive_request;

2.6.1 Provide Commercial Vehicle Manager Tag Data Interface

Input Flows

cf_tag_data_store_output
cf_tag_initialization_data
fcvm_carrier_number
fcvm_driver_number
fcvm_request_tag_data_output
fcvm_trip_identity
fcvm_vehicle_number

Output Flows

cf_tag_data
cf_tag_data_store_request
cf_tag_data_store_write
tcvm_confirm_enrollment_data_stored
tcvm_output_tag_data

Description:

Overview: This process shall be responsible for providing an interface through which the commercial vehicle manager can set up the data in the tag on-board a commercial vehicle. The data that the process enables the manager to write to the tag will be that which identifies the carrier, driver and vehicle. The process shall also enable the manager to read this data from the tag, but shall not enable the reading of any other data from the tag. Data provided by the manager shall also be sent by the process to the tag the process that manages electronic credentials and tax filing for use by the manager in future enrollments. The process shall support inputs from the commercial vehicle manager in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be in hardcopy, or as a display.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cf_tag_initialization_data';
(b) 'fcvm-carrier_number';
(c) 'fcvm-driver_number';
(d) 'fcvm-request_tag_data_output';
(e) 'fcvm-trip_identity';
(f) 'fcvm-vehicle_number'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'cf_tag_data_store_output';
(b) 'cf_clearance_enrollment_confirm'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cf_tag_data';
(b) 'cf_tag_data_store_request';
(c) 'cf_tag_data_store_write';
(d) 'tcvm-confirm_enrollment_data_stored';
(e) 'tcvm-output_tag_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above as originating from the commercial vehicle manager;
(b) when the inputs are received, generate the outputs identified above and write the data into the store identified above;
(c) also send the data to the process that manages electronic credentials and tax filing so that it can be used for future enrollments;
(d) use the appropriate mechanism(s) such as RDBMS, to write data to the store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.2;
USR = 4.1.2.2;

Output Flow Dynamics Assumptions:

cf_tag_data_store_write = 1/DAY;
cf_tag_data = 1/DAY;
tcvm-confirm_enrollment_data_stored = 1/(60*60*24);
tcvm-output_tag_data = fcvm-request_tag_data_output;
cf_tag_data_store_request = fcvm-request_tag_data_output;

2.6.2 Transmit Commercial Vehicle Tag Data

Input Flows
cv_lock_tag_data
cv_on_board_border_record
cv_on_board_screening_record
cv_request_electronic_clearance_data
cv_tag_data_store_read

Output Flows
cv_electronic_clearance_data
cv_request_lock_tag_data
cv_tag_data_store_needed
cv_tag_data_store_update

Description:
Overview: This process shall be responsible for providing the output of the data that has been previously stored on-board a commercial vehicle's tag on request from a commercial vehicle roadside checkstation facility. The process shall also provide the current status of the lock tag, if one is attached to the vehicle's cargo. The data shall only be sent by the process to the commercial vehicle roadside checkstation or border crossing facility that made the request. The output mechanism used by the process shall be an implementation issue, but it could be by radio, beacon, or a visual mechanism, such as a bar code.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_request_electronic_clearance_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of
output being sent to other processes and requests for data retrieval:
(a) 'cv_lock_tag_data' - which is data received from another process;
(b) 'cv_on_board_border_record' - which is data received from another process;
(c) 'cv_on_board_screening_record' - which is data received from another process;
(d) 'cv_tag_data_store_read' - which is data retrieved from a store.

Solicited Output Processing: This process shall provide the following output flows as a result of
the above inputs being received:
(a) 'cv_electronic_clearance_data';
(b) 'cv_request_lock_tag_data';
(c) 'cv_tag_data_store_needed':
(d) 'cv_tag_data_store_update'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input flow in (a) is received, de-encrypt the data and generate the solicited output
data flows shown above that request the current tag data and the current lock tag data;
(c) as a result of (b) await receipt of the solicited input data flow listed above containing the
data from the tag and de-encrypt it when it arrives;
(d) monitor for receipt of the first solicited data flow shown above (from the lock tag) and de-encrypt it when it arrives;
(e) if no data is received in (d) wait for a specified timeout period before concluding that none is
available and that the response data must be set to blank indicating no lock tag;
(f) when (c) and (d) have been completed, send the retrieved data to the requesting commercial
vehicle roadside checkstation or border crossing facility using the first solicited output data flow shown above;
(g) as a result of (e) monitor for receipt of either of the second or third solicited data flows
listed above, and when received, use the data they contain to generate the third solicited output data flow shown above;
(h) all solicited output data flows whether to the tag or the roadside facility must be encrypted
in such as way that it is not possible to determine their contents, using any digital or analog techniques.

User Service Requirements:
USR = 4.0;
USR = 4.1;
USR = 4.1.2;
USR = 4.1.2.1;
USR = 4.1.2.2;

Output Flow Dynamics Assumptions:
cv_electronic_clearance_data = cv_request_electronic_clearance_data;
cv_request_lock_tag_data = cv_request_electronic_clearance_data;
cv_tag_data_store_needed = cv_request_electronic_clearance_data;
cv_tag_data_store_update = cv_request_electronic_clearance_data;

2.6.3 Provide Commercial Driver Tag Data Interface

Input Flows

cv_tag_data_store_output
fcvd_carrier_number
fcvd_driver_number
fcvd_request_tag_data_output
fcvd_trip_identity
fcvd_vehicle_number

Output Flows

cv_tag_data_store_request
cv_tag_data_store_write
tcvd_confirm_data_stored
tcvd_output_tag_data

Description:

Overview: This process shall be responsible for providing the interface through which the commercial vehicle driver can set up the data in an on-board vehicle unit (e.g. an electronic tag). In this instance the driver is assumed to be acting in the role of a commercial vehicle manager, and is thus likely to be the owner of the vehicle. The data the process enables the manager to write to the tag will be that which identifies the carrier, driver and vehicle. The process shall also enable the read this data from the tag, but shall not enable the manager to read any other data from the tag. The process shall support inputs from the commercial vehicle driver in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be in hardcopy, or as a display.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fcvd-carrier_number';
(b) 'fcvd-driver_number';
(c) 'fcvd-request_tag_data_output';
(d) 'fcvd-trip_identity';
(e) 'fcvd-vehicle_number'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(a) 'cv_tag_data_store_request';
(b) 'cv_tag_data_store_write';
(c) 'tcvd-confirm_data_stored'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above as originating from the commercial vehicle driver;
(b) when the inputs are received, generate the outputs identified above and write the data into the store identified above;
(c) use the appropriate mechanism(s) such as RDBMS, to write data to the store identified above.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.2;
USR = 4.1.2.2;

Output Flow Dynamics Assumptions:

cv_tag_data_store_request = fcvd-request_tag_data_output;
cv_tag_data_store_write = 1/(60*60*24);
tcvd-confirm_data_stored = 1/(60*60*24);
tcvd-output_tag_data = fcvd-request_tag_data_output;

2.6.4 Provide Lock Tag Data Interface

Input Flows

cv_request_lock_tag_data
fcv_lock_tag_data

Output Flows

cv_lock_tag_data
tcv_lock_tag_data_request

Description:

Overview: This process shall be responsible for producing an output of the current status of a lock tag that is being carried by the cargo of a commercial vehicle. The process shall only produce the output in response to a request for data that is received from the other process on-board the vehicle that is responsible for communication with commercial vehicle roadside checkstation facilities. The actual output mechanism used by the process shall be an implementation issue, but it could be by radio or beacon.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_request_lock_tag_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to a terminator:
(a) 'fcv-lock_tag_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cv_lock_tag_data';
(b) 'tcv-lock_tag_data_request'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for the unsolicited input flow listed above;
(b) when the flow in (a) is received, output the second solicited output data flow shown above;
(c) continuously monitor for receipt of the solicited input flow shown above and when it is received, generate the first solicited output flow shown above;
(d) all solicited output data flows must be encrypted in such as way that it is not possible to determine their contents, using any digital or analog techniques.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.2;
USR = 4.1.2.2;

Output Flow Dynamics Assumptions:

cv_lock_tag_data = cv_request_lock_tag_data;
tcv-lock_tag_data_request = cv_request_lock_tag_data;

2.6.5 Manage Commercial Vehicle Tag Data Store

Input Flows

cf_tag_data_store_request
cf_tag_data_store_write
cv_tag_data_store
cv_tag_data_store_needed
cv_tag_data_store_request
cv_tag_data_store_update
cv_tag_data_store_write

Output Flows

cf_tag_data_store_output
cv_provide_credentials_data_for_inspections
cv_tag_data_store
cv_tag_data_store_output
cv_tag_data_store_read

Description:

Overview: This process shall be responsible for managing the store of data that is held by a commercial vehicle's on-board tag. It shall manage all of the transactions that either write data to the store and read data from it, to ensure that the data retains its consistency. The process shall ensure that the commercial vehicle manager or driver can only read the data that they are enabled to write to the store, and that the store only contains data from the last two roadside checkstation facilities passed by the commercial vehicle.

(a) 'cf_tag_data_store_request';
(b) 'cf_tag_data_store_write';
(c) 'cv_tag_data_store_needed';
(d) 'cv_tag_data_store_request';
(e) 'cv_tag_data_store_write';
(f) 'cv_tag_data_store_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval:
(a) 'cv_tag_data_store' - which is data retrieved from a store.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'cf_tag_data_store_output';
(b) 'cv_provide_credentials_data_for_inspections';
(c) 'cv_tag_data_store_output';
(d) 'cv_tag_data_store_read'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either of the input flows in (a) from the commercial vehicle manager or driver, containing data to be written to the tag is received, write the data it contains to the store, overwriting the same data already held in the store;
(c) as a result of (b) generate the flow that sends data for use in safety inspections;
(d) when the flow in (a) containing update data is received, write the data it contains to the store, overwriting the data from the oldest roadside facility that is already held in the store;
(e) when any of the data flows in (a) from the commercial vehicle manager or driver, requesting data is received, retrieve the current data from the store and generate the corresponding solicited output data flow shown above;
(f) the retrieved in (e) must not include the data sent by commercial vehicle roadside facilities;
(g) it must not be possible for the commercial vehicle manager or driver to over-write the data on the tag that has been sent from the roadside facilities;
(h) use the most appropriate mechanism such as RDBMS, to read and write data from and to the data store.

User Service Requirements:

USR = 4.0;
USR = 4.1;
USR = 4.1.2;
USR = 4.1.2.1;
USR = 4.1.2.2;

Output Flow Dynamics Assumptions:

cf_tag_data_store_output = cf_tag_data_store_request;
cv_provide_credentials_data_for_inspections = cf_tag_data_store_write + cv_tag_data_store_write;
cv_tag_data_store_output = cv_tag_data_store_request;
cv_tag_data_store_read = cv_tag_data_store_needed;

2.7 Manage Cargo

Input Flows

fifd_freight_data
From_Intermodal_Freight_Shipper

Output Flows

freight_cargo_data
tifd_freight_request
To_Intermodal_Freight_Shipper

Description:

Overview: This process shall be responsible for providing facilities for the management of cargo shipments. The process shall enable these shipments to be routed via intermodal shippers and depots and may not need the services of a commercial vehicle manager or driver.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received, generate the appropriate outputs.

User Service Requirements:

USR = 4.0;
USR = 4.2;
USR = 4.2.2;
USR = 4.2.2.7;
USR = 4.2.2.7(d);

Output Flow Dynamics Assumptions:

To_Intermodal_Freight_Shipper = From_Intermodal_Freight_Shipper;
tifd-freight_request = 3/(60*60*24);
freight_cargo_data = 3/(60*60*24);

3.1.1 Produce Collision and Crash Avoidance Data

Input Flows

collision_data
intersection_collision_avoidance_data

Output Flows

position_warnings
vehicle_action_requests

Description:

Overview: This process shall be responsible for sensing and evaluating the likelihood of a collision between two vehicles or a vehicle and a stationary object. The process shall base its detection on input from two other processes. One of these processes shall be that which continuously processes sensor inputs on-board the vehicle and the second shall be that which detects collision situations at intersections. When either event is detected this process shall output the appropriate messages to another process in the vehicle to warn the driver. If the vehicle is suitably equipped, the process shall initiate the deployment of crash restraint devices in advance of the collision and/or generate data to initiate direct operation of the vehicle to take evasive maneuvers.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when either of the input flows is received, generate the outputs identified above;
(c) the time to produce the outputs must be consistent with the safe operation of vehicle control systems with human interfaces regardless of the number of times any of the inputs are received;
(d) the content of the position warning message shall be tailored to reflect the data input received.

User Service Requirements:

USR = 6.0;
USR = 6.1;
USR = 6.1.0;
USR = 6.1.1;
USR = 6.1.1.2;
USR = 6.1.1.2.1;
USR = 6.1.2;
USR = 6.1.2.1;
USR = 6.1.2.2;
USR = 6.1.2.1.1;
USR = 6.1.2.2.1;
USR = 6.1.3;
USR = 6.1.3.1;
USR = 6.1.3.1.1;
USR = 6.1.3.2;
USR = 6.1.3.2.1;
USR = 6.1.3.3;
USR = 6.1.3.3.1;
USR = 6.2;
USR = 6.2.0;
USR = 6.2.1;
USR = 6.2.1.1;
USR = 6.2.1.1.1;
USR = 6.2.1.2;
USR = 6.2.1.2.2;
USR = 6.2.2;
USR = 6.2.2.1;
USR = 6.2.2.1.1;
USR = 6.3;
USR = 6.3.0;
USR = 6.3.1;
USR = 6.3.1.1;
USR = 6.3.2;
USR = 6.3.2.1;
USR = 6.5;
USR = 6.5.1;
USR = 6.5.1.1;
USR = 6.5.1.1.2;
USR = 6.6;
USR = 6.6.1;

USR = 6.6.1.1;
USR = 6.6.1.2;

Output Flow Dynamics Assumptions:

position_warnings = 1/(60*60*24);
vehicle_action_requests = 1/(60*60*24);

3.1.2 Carry-out Safety Analysis

Input Flows

safety_data

Output Flows

safety_warnings
vehicle_and_driver_safety_status

Description:

Overview: This process shall be responsible for producing safety warnings for display to the driver and output to the vehicle control processes. The process shall base its output on input from another process in the vehicle that is analyzing inputs to sensors. When data about a safety situation is received, the process shall output the appropriate messages to another process in the vehicle to warn the driver. If the vehicle is so equipped, the process shall send data to the process in the vehicle responsible for its control.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the outputs identified above;
(c) the time to produce the outputs must be consistent with the safe operation of vehicle control systems with human interfaces regardless of the number of time the input is received.
(d) only send the data to the vehicle control processes if the input shows that there is a safety problems with the driver or the vehicle;
(e) the content of the safety warning message shall be tailored to reflect the data input received.

User Service Requirements:

USR = 6.0;
USR = 6.5;
USR = 6.5.1;
USR = 6.5.1.1;
USR = 6.5.1.1.1;
USR = 6.5.1.1.2;
USR = 6.5.1.1.3;
USR = 6.1.2;
USR = 6.1.2.1;
USR = 6.5.2.1.1;
USR = 6.5.2.1.2;
USR = 6.0;
USR = 6.5;
USR = 6.5.3;
USR = 6.5.3.1;
USR = 6.5.3.1.1;
USR = 6.5.3.1.2;
USR = 6.7;
USR = 6.7.0;
USR = 6.7.1;
USR = 6.7.1.3;
USR = 6.7.1.3.1;

Output Flow Dynamics Assumptions:

safety_warnings = 1/(60*60);
vehicle_and_driver_safety_status = 1/(60*60);

3.1.3 Process Vehicle On-board Data

Input Flows

fbv_diagnostics_data
fbv_driver_safety_status
fbv_vehicle_attitude_data
fbv_vehicle_motion_data
fbv_vehicle_proximity_data
fbv_vehicle_safety_status
fbv_vehicle_security_status
fre_roadside_data
From_Potential_Obstacles
From_Roadway

Output Flows

collision_data
safety_data
vehicle_smart_probe_data
vehicle_status_details_for_broadcast
vehicle_status_details_for_driver_security
vehicle_status_details_for_emergencies
vehicle_status_details_for_emissions

Description:

Overview: This process shall be responsible for processing data received as input to sensors located on-board a vehicle. The process shall continuously analyze these inputs and produce data from which safety and/or position warnings and actions can be produced by another process. It shall also analyze the data to check for hazardous roadside conditions such as flooding, ice, snow, etc. and if detected shall output this data to processes in the Manage Traffic function.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously generate the outputs list above, using data scanned from the inputs also listed above;
(b) complete a full scan of all inputs and generate the outputs in a timeframe consistent with the safe operation of vehicle control systems regardless of the number of inputs to be scanned and the number of outputs generated;
(c) the vehicle probe data shall contain details of the type of hazard found on the road around the vehicle, or be blank of there are none;
(d) be capable of accepting input data in a variety of formats, both digital and analog.

User Service Requirements:

USR = 1.6;
USR = 1.6.4;
USR = 1.6.4(a);
USR = 6.0;
USR = 6.1;
USR = 6.1.0;
USR = 6.1.1;
USR = 6.1.1.1;
USR = 6.1.1.1.1;
USR = 6.1.1.1.1.1;
USR = 6.1.1.1.1.2;
USR = 6.1.1.1.2;
USR = 6.1.1.2;
USR = 6.1.1.2.1;
USR = 6.1.1.3;
USR = 6.1.1.3.1;
USR = 6.1.2;
USR = 6.1.2.1;
USR = 6.1.2.1.1;
USR = 6.1.2.2.1;
USR = 6.1.2.3.1;
USR = 6.2;
USR = 6.2.0:
USR = 6.2.1;
USR = 6.2.1.1;

USR = 6.2.1.1.1;
USR = 6.2.1.2;
USR = 6.2.1.2.2;
USR = 6.2.1.3;
USR = 6.2.1.3.1;
USR = 6.2.2;
USR = 6.2.2.1;
USR = 6.2.2.1.1;
USR = 6.2.2.2;
USR = 6.2.2.2.1;
USR = 6.2.2.3;
USR = 6.2.2.3.1;
USR = 6.3;
USR = 6.3.0;
USR = 6.3.1;
USR = 6.3.1.1;
USR = 6.3.2;
USR = 6.3.2.1;
USR = 6.3.3;
USR = 6.3.3.1;
USR = 6.5;
USR = 6.5.1;
USR = 6.5.1.1;
USR = 6.5.1.1.1
USR = 6.5.1.1.2;
USR = 6.5.1.1.3;
USR = 6.5.2;
USR = 6.5.2.1;
USR = 6.5.2.1.1;
USR = 6.5.2.1.2;
USR = 6.5.3;
USR = 6.5.3.1;
USR = 6.5.3.1.1;
USR = 6.6;
USR = 6.6.1;
USR = 6.6.1.1;
USR = 6.6.1.2;
USR = 6.7;
USR = 6.7.2;
USR = 6.7.2.1;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:;

collision_data = 1;
safety_data = 1;
vehicle_smart_probe_data = 1;
vehicle_status_details_for_broadcast = 1;
vehicle_status_details_for_driver_security = 1;
vehicle_status_details_for_emergencies = 1;
vehicle_status_details_for_emissions = 1;

3.2.1 Provide Driver Interface

Input Flows

ahs_status
control_status
vehicle_control_request

Output Flows

driver_ahs_input
driver_input
vehicle_control_status

Description:

Overview: This process shall be responsible for providing an interface through which a vehicle driver can initiate, monitor and terminate automatic control of the vehicle. The output that any of these actions generates in terms of messages to the driver shall be sent by this process to another process that is in the Provide Driver and Traveler Services function and in the vehicle. The driver inputs shall be received by this process from another process that is also in the Provide Driver and Traveler Services function and in the vehicle.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) the time to produce the outputs must be consistent with the safe operation of an ahs vehicle control system regardless of the number of times the any of the inputs are received;
(d) interpretion of the vehicle control data flow shall be as shown in the flow definition;
(e) if ahs lane use has been activated and a request for platooning, speed control, headway control or lane control is detected in the vehicle control data flow, the flow to the ahs control process shall be sent with manual mode requested;
(f) when the response to (e) shows that the vehicle is leaving the ahs lanes, then data for the requested automatic control mode shall be sent to the processes responsible for this mode of vehicle operation;
(g) if the control status input shows any of the failure modes, the flow to the ahs control process shall be sent with manual mode requested;
(h) all changes in mode of vehicle operation shall be accompanied by the output of appropriate warning data to the processes that provide driver outputs;
(i) all inputs for changes in vehicle control mode must be asserted for a specified amount of time before they are interpreted;
(j) all inputs for changes in control status must be present for a specified amount of time.

User Service Requirements:

USR = 6.0;
USR = 6.6;
USR = 6.6.0;
USR = 6.6.1;
USR = 6.6.1.1;
USR = 6.6.1.2;

Output Flow Dynamics Assumptions:

driver_ahs_input = 2/(60*60*24);
driver_input = 1/(60*60);
vehicle_control_status = 1;

3.2.2 Provide AHS Control

Input Flows

ahs_check_response
ahs_route
ahs_vehicle_data
driver_ahs_input
platoon_status

Output Flows

ahs_control_data_update
ahs_route_data
ahs_route_request
ahs_status
ahs_vehicle_condition
platoon_action

Description:

Overview: This process shall be responsible for providing the facility that enables vehicles to operate in automatic highway system (ahs) lanes. This mode of operation shall only be initiated by the process when a request is received from the driver via other processes in the vehicle. The first action of the process must be to send data to the process that provides the ahs check-in facility. If a positive response is received from that process, i.e. the vehicle's check in is accepted, then the process shall enable ahs operation by sending the data to the vehicle control processes. Once the vehicle is in ahs operation, the process shall continuously monitor for an input from the driver that cancels ahs mode, and when this is received send mode canceling data to the vehicle control processes. Similarly the process shall also continuously monitor input from the process analyzing vehicle condition and the vehicle's presence on an ahs lane. The process shall send mode canceling data to the vehicle control processes, if the condition does not support ahs lane operation, or the vehicle is no longer on an ahs lane.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the driver ahs input data flow is are received, implement the instructions it contains;
(c) if ahs mode is requested in (b) and the vehicle is not in that mode, initiate the ahs check-in procedure, and when successfully completed, the ahs route generation procedure, and when this is completed, initiate ahs control by sending data to the vehicle control processes;
(d) if the ahs check-in procedure fails then set the ahs status output to check-in failed and do not proceed further with any ahs processing or send any control parameters to other processes;
(e) when the ahs based route has been selected, send the list of route segments to the process that manages ahs operations for loading into the operational data store;
(f) if the vehicle condition input shows that the vehicle cannot continue in ahs mode or the driver ahs input requests departure from ahs mode, send the ahs cancel data to the vehicle control processes before generating any other outputs;
(g) any change in ahs status shall produce an ahs status data flow output;
(h) this process must scan all inputs and complete the subsequent processing in a time frame consistent with the safe operation of ahs vehicle control systems regardless of the number of changes in input and any interruptions.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.1;
USR = 6.7.1.1.1;
USR = 6.7.1.1.2;
USR = 6.7.1.1.3;
USR = 6.7.1.2;
USR = 6.7.1.2.1;
USR = 6.7.1.2.3;
USR = 6.7.1.3;
USR = 6.7.1.3.2;
USR = 6.7.2;
USR = 6.7.2.1;
USR = 6.7.2.2;

Output Flow Dynamics Assumptions:

```
ahs_control_data_update = driver_ahs_input;
ahs_route_data = driver_ahs_input;
ahs_route_request = driver_ahs_input;
ahs_status = 1;
ahs_vehicle_condition = driver_ahs_input;
platoon_action = 1;
```

3.2.3.1 Provide Command Interface

Input Flows

driver_input
feedback_actuator_status
feedback_platoon_status
feedback_sensor_status
feedback_servo_status

Output Flows

control_status
driver_commands
driver_manual_input
driver_selection

Description:

Overview: This process shall be responsible for providing the interface through which all driver commands are passed to the correct processes in the vehicle for action. The process shall also pass all messages about vehicle control status on to another process in the vehicle for output to the driver. It shall also monitor the health of the other in-vehicle processes involved in automatic vehicle control. This process shall take the appropriate mode canceling action when any failures are detected in these processes.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor the platoon status feedback data flow and if at any time it shows that the vehicle and/or driver is unsafe or that a built in self test (bist) failure has been detected, send unsafe or failure data messages to the processes responsible for the driver interface, platoon following, vehicle actuators and servo actuators;
(b) if the platoon status feedback data flow is set to safe, continuously monitor the data flow driver input for any changes;
(c) if the flow in (b) changes from manual input detected to any other state then if none of the feedback status messages are set to failure send data to the appropriate processes dependent on the mode of automatic vehicle control requested by the driver;
(d) following (c) monitor the return data from the receiving process(es) and pass on the data to the driver interface process, setting the output to failure if no response is received within a specified time-out period from the time the data is being sent;
(e) all activity by this process must be completed within a time frame consistent with safe operation of ahs vehicle control systems regardless of the activity involved, or the number of inputs being processed.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.1;
USR = 6.7.1.2.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.1

Output Flow Dynamics Assumptions:

control_status = 1;
driver_commands = 1/(60*60);
driver_manual_input = 1/(60*60);
driver_selection = 1/(60*60);

3.2.3.2 Manage Platoon Following

Input Flows

data_from_front_vehicle
data_from_rear_vehicle
driver_selection
manual_input_received
platoon_action
vehicle_and_driver_safety_status
vehicle_control_data

Output Flows

data_to_front_vehicle
data_to_rear_vehicle
feedback_platoon_status
platoon_following_commands
platoon_status
platooning_selected

Description:

Overview: This process shall be responsible for providing the facility for the automatic control of vehicles to be extended to cover the platooning of vehicles. The process shall enable vehicles to follow each other very closely (inches apart) in a platoon, responding to changes in speed and direction of the lead vehicle. The process shall monitor data from other vehicles in the platoon received via another process, and shall also send data about itself to the same process for communication to other platoon vehicles. If the data received from the process shows that the vehicle has been left on its own, i.e. there are no other vehicles in front or behind, the process shall send data to another process in the vehicle to increase speed and catch up with any platoon that may be ahead. The process shall only allow the vehicle to join or continue running in a platoon if it and/or the driver are considered to be in a safe condition, using data received from other processes in the vehicle.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above so that the vehicle maintains the required platoon following status in a manner that ensures its safety and that of other vehicles regardless of whether the vehicles behind or in front are joining or leaving the platoon;
(c) if requested to leave the platoon, do so in a manner that ensures the safe operation of the remaining vehicles in the platoon and any other vehicles on the highway, as well as the safety of the driver plus other vehicle occupants;
(d) all activity by this process must be completed within a time frame consistent with the safe operation of vehicle control systems regardless of the activity involved, or the number of inputs being processed.

User Service Requirements:

USR = 6.0;
USR = 6.1;
USR = 6.1.1;
USR = 6.1.1.1;
USR = 6.1.1.1.1;
USR = 6.1.1.1.2.1;
USR = 6.1.1.1.2.2;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

data_to_front_vehicle = 1*AHS_VEHS;
data_to_rear_vehicle = 1*AHS_VEHS;
feedback_platoon_status = 1;
platoon_following_commands = 1;
platooning_selected = 1;
platoon_status = 1;

3.2.3.3 Process data for Vehicle Actuators

Input Flows

actuator_commands
driver_manual_input
fbv_brake_servo_response
fbv_steering_servo_response
fbv_throttle_servo_response
platooning_selected
vehicle_action_requests

Output Flows

feedback_actuator_response
feedback_actuator_status
manual_input_received
tbv_change_brake_setting
tbv_change_direction
tbv_change_throttle_setting
tbv_deploy_crash_restraints
tbv_steer_left
tbv_steer_right
tbv_steer_straight

Description:

Overview: This process shall be responsible for providing the interface between other automatic vehicle control process and the actuators which actually change the vehicle's controls. The process shall both implement commands and monitor the operation of the actuators to check that they only move when requested. If they move for any other reason, e.g. the driver has touched the vehicle controls, the process shall disable automatic operation. The process shall perform its own built-in self test (BIST) analysis. It shall report any errors that this shows to another process in the vehicle and shall cease to accept further requests to change the vehicle's actuators.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) respond immediately to requests for any changes in actuator state regardless of the source of the request;
(c) if any actuator does not make the requested change, or moves without input, send an error response to the process responsible for providing servo controls, provide command interface and platoon following;
(d) regardless of any other actions being carried out, continuously perform an operational self check using a watchdog to ensure that all actions are being carried out in time and that the processor is functioning correctly, and if a failure occurs set the BIST result in the data flow sent to the process responsible for monitoring actuator status and providing the command interface;
(e) the process shall assume manual mode of operation on start up and until instructed by input flows to change to any other condition.

User Service Requirements:

USR = 6.0;
USR = 6.1;
USR = 6.1.1;
USR = 6.1.1.1;
USR = 6.1.1.1.1;
USR = 6.1.1.1.2;
USR = 6.1.1.1.2.1;
USR = 6.1.1.1.2.2;
USR = 6.1.1.1.3;
USR = 6.1.1.1.3.1;
USR = 6.1.1.3;
USR = 6.1.1.3.1;
USR = 6.1.2;
USR = 6.1.2.3;
USR = 6.1.2.3.1;
USR = 6.1.3
USR = 6.1.3.3
USR = 6.1.3.3.1;

USR = 6.2;
USR = 6.2.1;
USR = 6.2.1.3;
USR = 6.2.1.3.1
USR = 6.2.2;
USR = 6.2.2.3;

Output Flow Dynamics Assumptions:

feedback_actuator_response = 1;
feedback_actuator_status = 1;
manual_input_received = 1; ;
tbv-change_brake_setting = 1;;
tbv-change_direction = 1;
tbv-change_throttle_setting = 1;
tbv-deploy_crash_restraints = 1;
tbv-steer_left = 1;
tbv-steer_right = 1; ;
tbv-steer_straight = 1;

3.2.3.4.1 Provide Speed Servo Control

Input Flows

manual_throttle_input_detected
override_throttle
platoon_speed_servo_override
select_speed
speed
vehicle_speed_control_data

Output Flows

feedback_speed_servo_status
throttle_commands

Description:

Overview: This process shall be responsible for providing data which enables the vehicle's throttle to be regulated in such a way that a desired vehicle speed is maintained. The process shall enable the throttle to be overridden temporarily in order to maintain a desired headway between the vehicle and others in a platoon. The data that actually changes the throttle's position shall be sent to the process that provides data to in-vehicle actuators. The process shall perform its own built-in self test (BIST) analysis. It shall report any errors that this shows to another process in the vehicle and shall cease to accept further requests to change the vehicle's throttle position.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above to either maintain the requested speed or temporarily change to another value for as long as the override data input is present;
(c) the output of the new throttle setting is to be sent to the vehicle control data interface process for onward communication to the process responsible for vehicle actuators;
(d) regardless of any other actions being carried out, continuously perform an operational self check using a 'watchdog' to ensure that all actions are being carried out in time and that the processes is functioning correctly;
(e) if a failure occurs in (d) then set the BIST result part of the speed servo status feedback data flow to indicate a failure and send it to the process that provides the command interface;
(f) the time to produce the outputs must be consistent with the safe operation of vehicle control systems regardless of the number of times the any of the inputs are received.

User Service Requirements:

USR = 6.0;
USR = 6.1;
USR = 6.1.1;
USR = 6.1.1.1;
USR = 6.1.1.1.1;
USR = 6.1.1.1.2;
USR = 6.1.1.1.2.1;
USR = 6.1.1.1.2.2;
USR = 6.1.1.1.3;
USR = 6.1.1.1.3.1;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

feedback_speed_servo_status = 1;
throttle_commands = 1;

3.2.3.4.2 Provide Headway Servo Control

Input Flows

headway
manual_brake_input_detected
platoon_headway_servo_override
select_headway
vehicle_headway_control_data

Output Flows

brake_commands
feedback_headway_servo_status
override_throttle

Description:

Overview: This process shall be responsible for providing data which enables the vehicle's brake and throttle to be regulated in such a way that its headway, i.e. the distance between it and the vehicle in front, is maintained. The process shall support the brake movements that either maintain the vehicle's headway for normal operation, or hold it at the value used in platoon following, whether on or off automated highway system (ahs) lanes. The process shall perform its own built-in self test (BIST) analysis. It shall report any errors that this shows to another process in the vehicle and shall cease to accept further requests to change the vehicle's brake setting.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above to either maintain the requested headway or temporarily change to another value for as long as the override data input is present;
(c) the output of the new brake setting is to be sent to the vehicle control data interface process for onward communication to the process responsible for vehicle actuators;
(d) regardless of any other actions being carried out, continuously perform an operational self check using a 'watchdog' to ensure that all actions are being carried out in time and that the processes is functioning correctly;
(e) if a failure occurs in (d) then set the BIST result part of the speed servo status feedback data flow to indicate a failure and send it to the process that provides the command interface;
(f) the time to produce the outputs must be consistent with the safe operation of vehicle control systems regardless of the number of times the any of the inputs are received.

User Service Requirements:

USR = 6.0;
USR = 6.1;
USR = 6.1.1;
USR = 6.1.1.1;
USR = 6.1.1.1.1;
USR = 6.1.1.1.2;
USR = 6.1.1.1.2.1;
USR = 6.1.1.1.2.2;
USR = 6.1.1.1.3;
USR = 6.1.1.1.3.1;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

brake_commands = 1;
feedback_headway_servo_status = 1;
override_throttle = 1;

3.2.3.4.3 Provide Lane Servo Control

Input Flows

lane_deviation
manual_steering_input_detected
override_lane_hold
platoon_lane_servo_override
select_lane_hold

Output Flows

feedback_lane_servo_status
lane_steering_commands

Description:

Overview: This process shall be responsible for providing the data which enables the vehicle's steering to be adjusted so that it maintains a position that is in the middle of its current lane. The process shall enable this to be temporarily overridden as a result of action being taken by other processes to change lanes. The process shall perform its own built-in self test (BIST) analysis. It shall report any errors that this shows to another process in the vehicle and shall cease to accept further requests to change the vehicle's throttle position.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above to either maintain the requested lane position or temporarily change to another lane for as long as the override data input is present;
(c) the output of the lane position setting is to be sent to the vehicle control data interface process for onward communication to the process responsible for vehicle actuators;
(d) regardless of any other actions being carried out, continuously perform an operational self check using a 'watchdog' to ensure that all actions are being carried out in time and that the processes is functioning correctly;
(e) if a failure occurs in (d) then set the BIST result part of the speed servo status feedback data flow to indicate a failure and send it to the process that provides the command interface;
(f) the time to produce the outputs must be consistent with the safe operation of vehicle control systems regardless of the number of times the any of the inputs are received.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

feedback_lane_servo_status = 1;
lane_steering_commands = 1;

3.2.3.4.4 Provide Change Lane Servo Control

Input Flows

manual_steering_input_detected
platoon_change_lane_servo_override

Output Flows

feedback_change_lane_servo_status
override_lane_hold
steering_commands

Description:

Overview: This process shall be responsible for providing the data which enables the vehicle's steering to be adjusted so that it will move either left or right from one lane to another. The process shall enable this to temporarily override the lane center holding facility available from another process in the vehicle. The process shall perform its own built-in self test (BIST) analysis. It shall report any errors that this shows to another process in the vehicle and shall cease to accept further requests to change the vehicle's throttle position.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above to change from one lane to another for as long as the input is received;
(c) the output of the lane change data is to be sent to the vehicle control data interface process for onward communcation to the process responsible for vehicle actuators;
(d) regardless of any other actions being carried out, continuously perform an operational self check using a 'watchdog' to ensure that all actions are being carried out in time and that the processes is functioning correctly;
(e) if a failure occurs in (d) then set the BIST result part of the speed servo status feedback data flow to indicate a failure and send it to the process that provides the command interface;
(f) the time to produce the outputs must be consistent with the safe operation of vehicle control systems regardless of the number of times the any of the inputs are received.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

feedback_change_lane_servo_status = 1;
override_lane_hold = 1;
steering_commands = 1;

3.2.3.4.5 Provide Vehicle Control Data Interface

Input Flows

ahs_control_data_update
brake_commands
driver_commands
feedback_actuator_response
feedback_change_lane_servo_status
feedback_headway_servo_status
feedback_lane_servo_status
feedback_speed_servo_status
lane_steering_commands
platoon_following_commands
sensor_data
steering_commands
throttle_commands
vehicle_control_data_store

Output Flows

actuator_commands
feedback_servo_status
headway
lane_deviation
manual_brake_input_detected
manual_steering_input_detected
manual_throttle_input_detected
platoon_change_lane_servo_override
platoon_headway_servo_override
platoon_lane_servo_override
platoon_speed_servo_override
select_headway
select_lane_hold
select_speed
speed
vehicle_control_data
vehicle_headway_control_data
vehicle_speed_control_data

Description:

Overview: This process shall be responsible for providing a communications and data processing interface between processes in the Provide Vehicle Control and Monitoring function. These processes shall comprise those responsible for controlling individual functions, e.g. throttle, brake, etc., and those that interface to actuators and those that monitor vehicle operation.

Data Flows: All input data flows are unsolicited and all output flows are solicited.
Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) the time to produce the outputs should be consistent with the safe operation of vehicle control systems regardless of the number of times the any of the inputs are received.

User Service Requirements:

USR = 6.1.0;
USR = 6.1.1;
USR = 6.1.1.1;
USR = 6.1.1.1.1;
USR = 6.1.1.1.2;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.2;
USR = 6.7.1.2.3;
USR = 6.7.2;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

actuator_commands = 1;
feedback_servo_status = 1;
headway = 1;
lane_deviation = 1;

```
manual_brake_input_detected = 1;
manual_throttle_input_detected = 1;
manual_steering_input_detected = 1;
platoon_change_lane_servo_override = 1;
platoon_headway_servo_override = 1;
platoon_lane_servo_override = 1;
```

3.2.3.5 Process Vehicle Sensor Data

Input Flows

fbv_vehicle_headway
fbv_vehicle_lane_position
fbv_vehicle_on_ahs_lane
fbv_vehicle_speed

Output Flows

feedback_sensor_status
sensor_data

Description:

Overview: This process shall be responsible for providing the facility to decode the input being sent to on-board vehicle sensors. The process shall support inputs to those sensors that monitor conditions both on-board the vehicle and in the way the vehicle relates to its surroundings. The data produced by the process shall be sent to another process which shall determine if any action is required.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) convert the inputs from whatever form they are in (analog, digital, etc.) into a digital output which can be used by other processes;
(c) send the outputs periodically to the receiving processes regardless of the state of the inputs;
(d) this process shall complete its activity in within the period specified in (c) regardless of the number of inputs being processed, or the amount of processing needed to convert the inputs into digital signals.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.1;
USR = 6.7.1.1.3;
USR = 6.7.1.2;
USR = 6.7.1.2.1;
USR = 6.7.1.2.3;
USR = 6.7.1.3;
USR = 6.7.1.3.2;
USR = 6.7.2;
USR = 6.7.2.1;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

feedback_sensor_status = 1;
sensor_data = 1;

3.2.3.6 Communicate with other Platoon Vehicles

Input Flows

data_to_front_vehicle
data_to_rear_vehicle
From_Other_Vehicle

Output Flows

data_from_front_vehicle
data_from_rear_vehicle
To_Other_Vehicle

Description:

Overview: This process shall be responsible for communicating with the other vehicles that are in a platoon. The process shall support communications with the platoon vehicles that are both immediately in front of and behind the vehicle in which it operates. The passing of data in both directions, i.e. both to and from the vehicles, shall be supported by the process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs to other processes within a time frame consistent with the safe operation of vehicle control systems regardless of the number of inputs received at any one instant;
(c) data that is received from processes within the vehicle for output to other vehicles shall also meet the processing criteria in (b) above;
(d) if not data is received in (c) then do not send data to any other vehicle.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.1;
USR = 6.7.1.1.3;
USR = 6.7.1.2;
USR = 6.7.1.2.1;
USR = 6.7.1.2.3;
USR = 6.7.1.3;
USR = 6.7.1.3.2;
USR = 6.7.2;
USR = 6.7.2.1;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

data_from_front_vehicle = From_Other_Vehicle;
data_from_rear_vehicle = From_Other_Vehicle;
To_Other_Vehicle = data_to_front_vehicle+data_to_rear_vehicle;

3.2.4 Process Sensor Data for AHS input

Input Flows

fbv_vehicle_condition
fbv_vehicle_on_ahs_lane

Output Flows

ahs_vehicle_data

Description:

Overview: This process shall be responsible for analyzing the input from the vehicle that provides information about its condition and that it is on an automatic highway system (ahs) lane. The process shall continuously analyze this data and provide output to the process that provides ahs control.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) convert the inputs from whatever form they are in (analog, digital, etc.) into digital form that can be used by other processes;
(c) complete the processing required in (b) and the output of the resultant data in a time frame consistent with the safe operation of vehicle control systems, regardless of the number of inputs detected and the amount of processing needed to convert them into digital outputs.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.1;
USR = 6.7.1.1;
USR = 6.7.1.1.3;
USR = 6.7.1.2;
USR = 6.7.1.2.1;
USR = 6.7.1.2.3;
USR = 6.7.1.3;
USR = 6.7.1.3.2;
USR = 6.7.2;
USR = 6.7.2.1;
USR = 6.7.2.3;

Output Flow Dynamics Assumptions:

ahs_vehicle_data = 1;

3.2.5 Check Vehicle for AHS eligibility

Input Flows

ahs_control_information
ahs_vehicle_checking_parameters
ahs_vehicle_condition

Output Flows

ahs_check_response
ahs_checking_data
ahs_vehicle_checking_parameters

Description:

Overview: This process shall be responsible for checking that vehicles are eligible for using the automated highway system (ahs) lanes on a highway. The process shall decide whether or not the vehicle is suitable for has operation by checking locally stored data that has been provided by a process in the Manage Traffic function, against data from the vehicle provided through the check request by a process on-board the vehicle. The process shall send the results of the check to the process on-board the vehicle that requested the ahs check-in. The vehicles that are successfully checked-in shall also be down loaded with ahs control data from this process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ahs_vehicle_condition'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'ahs_control_information';
(b) 'ahs_vehicle_checking_parameters'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ahs_check_response';
(b) 'ahs_checking_data';
(c) 'ahs_vehicle_checking_parameters'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) check the data obtained from the vehicle against the locally stored parameters to establish that the vehicle is eligible for ahs operation;
(c) if a vehicle satisfies the parameter check, return a positive response in the first output flow listed above and include in it the special vehicle control data used for ahs operation;
(d) if a vehicle fails the parameter check, only send back a negative response in the first output flow listed above;
(e) be responsible for the management of the data in the store of the parameters against which vehicles are checked for ahs operation, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.0;
USR = 6.7.1;
USR = 6.7.1.1;
USR = 6.7.1.1.2;

Output Flow Dynamics Assumptions:

ahs_check_response = ahs_vehicle_condition;
ahs_checking_data = ahs_vehicle_condition;

3.2.6 Manage AHS Check-in and Check-out

Input Flows

ahs_checking_data
ahs_control_data_changes
ahs_route_data

Output Flows

ahs_checking_details
ahs_control_information

Description:

Overview: This process shall be responsible for managing the checking in and checking out of suitably equipped vehicles requesting to use automated highway system (ahs) lanes. The process shall provide the special vehicle control parameters needed for ahs operation to the process that manages ahs check-in and collect data on vehicles that request check-in and check-out from that process. This process shall send a record of all check-in and check-out transactions regardless of whether they are successful or not, to the process responsible for managing ahs operational data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ahs_control_data_changes';
(b) 'ahs_checking_data';
(c) 'ahs_route_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ahs_checking_details';
(b) 'ahs_control_information'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) store the input data identified above when ever it is received, and send the first output flow shown above back to the process responsible for checking vehicles for ahs operation.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.0;
USR = 6.7.1;
ahs_vehicle_checking_parameters = ahs_control_information; USR = 6.7.1.1;
USR = 6.7.1.1.1;
USR = 6.7.1.1.2;
USR = 6.7.1.1.3;

Output Flow Dynamics Assumptions:

ahs_checking_details = ahs_checking_data;
ahs_control_information = ahs_checking_data;

3.2.7 Manage AHS Operations

Input Flows

ahs_checking_details
ahs_control_data
ahs_usage_data

Output Flows

ahs_control_data_changes
ahs_operational_data
ahs_usage_data

Description:

Overview: This process shall be responsible for recording data about vehicles that have requested check-in and check-out for the use of the automated highway system (ahs) lanes, and for receiving ahs control parameters from a process in the Manage Traffic function. The process shall provide a process at the roadside with the vehicle control parameters needed for ahs operation. The process shall keep a log of all ahs check-in and check-out transactions received from the roadside process regardless of whether they are successful or not, and periodically pass this data on to the Plan System Deployment function.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'ahs_control_data';
(b) 'ahs_checking_details'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from a local data store:
(a) 'ahs_usage_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ahs_control_data_changes';
(b) 'ahs_operational_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) store the input data identified above when ever it is received, and send the first output flow shown above to the process responsible for checking vehicles for ahs operation;
(c) periodically read the data from the ahs transaction store and send it to the Plan System Deployment function using the second output flow listed above;
(d) be responsible for the management of the data in the store containing the ahs transaction log, using the appropriate mechanism(s) such as RDBMS, for storing the data.

User Service Requirements:

USR = 6.0;
USR = 6.7;
USR = 6.7.0;
USR = 6.7.1;
USR = 6.7.1.1;
USR = 6.7.1.1.1;
USR = 6.7.1.1.2;
USR = 6.7.1.1.3;

Output Flow Dynamics Assumptions:

ahs_control_data_changes = ahs_checking_data;
ahs_operational_data = 1/(60*60*24);

3.3.1 Provide Cargo Data for Incident Notification

Input Flows

cargo_data_request
fcv_cargo_data
fcv_cargo_safety_status

Output Flows

processed_cargo_data

Description:

Overview: This process shall be responsible for providing data about a commercial vehicle's cargo in the event that the vehicle is involved in some type of emergency. The process shall produce the output on request from another process in the vehicle regardless of whether the cargo has itself suffered from any damage. The cargo data being provided by the process shall cover all types, regardless of whether or not they are classified as HAZMAT cargoes.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow requesting cargo status listed above;
(b) when the input in (a) is are received, generate the outputs using data collected by processing the inputs from the on-board commercial vehicle sensors identified above;
(c) the processing of the sensor input and generation of the output shall be completed within a time frame consistent with the safe operation of vehicle control systems regardless of the number of inputs and the amount of processing needed to produce a digital output that can be used by other processes.

User Service Requirements:

USR = 4.0;
USR = 4.5;
USR = 4.5.0;
USR = 4.5.1;
USR = 4.5.1;
USR = 4.5.1.1;
USR = 4.5.1.2;
USR = 4.5.1.2(a);
USR = 4.5.1.2(b);
USR = 4.5.1.2(c);
USR = 5.0;
USR = 5.1;
USR = 5.1.2;
USR = 5.1.2.1;
USR = 5.1.2.1.2;
USR = 5.1.2.2;
USR = 5.1.2.2(c);

Output Flow Dynamics Assumptions:

processed_cargo_data = 1/(60*60*24*7*52);

3.3.2 Provide Communications Function

Input Flows

emergency_data_request
emergency_request_vehicle_acknowledge
vehicle_emergency_request

Output Flows

emergency_message_auto_output
emergency_request_vehicle_details
tbv_vehicle_security_system_commands

Description:

`

Overview: This process shall be responsible for sending messages it receives from other processes in this facility to the Manage Emergency Services function. It shall also be responsible for passing on the resulting response to the driver via processes in the Provide Driver and Traveler Services function. this process is also capable of receiving requests for additional data from the Manage Emergency Services function and transmitting follow-up details. this process can also receive commands related to the vehicle's security system from the Manage Emergency Services function and forward the commands to the vehicle's security system.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the vehicle emergency request flow listed above;
(b) when the input is received, immediately generate the output to the Manage Emergency Services function identified above;
(c) when a response is received to the output in (b) send the data flow to the Provide Driver and Traveler Services function so that a message can be output to the driver;
(d) the processing of the sensor input and generation of the output shall be completed within a time frame consistent with the safe operation of vehicle control systems regardless of the number of inputs and the amount of processing needed to produce a digital output that can be used by other processes.

User Service Requirements:

USR = 5.0;
USR = 5.1;
USR = 5.1.1;
USR = 5.1.1.3;

Output Flow Dynamics Assumptions:

emergency_message_auto_output = 1/YEAR;
emergency_request_vehicle_details = 1/YEAR*ITS_PVT_VEHS+1/YEAR*ITS_CVO_VEHS;
tbv-vehicle_security_system_commands = 1/YEAR;

3.3.3 Build Automatic Collision Notification Message

Input Flows

fbv_crash_sensor_data
processed_cargo_data
vehicle_identity_for_collision_notification_store
vehicle_location_for_incidents
vehicle_status_details_for_emergencies

Output Flows

cargo_data_request
vehicle_emergency_request

Description:

Overview: This process shall be responsible for preparing and submitting data for transmission to the Manage Emergency Services function. The data shall be sent by this process when an emergency situation is detected by analyzing inputs from the vehicle. This process shall produce its outputs regardless of any action by the driver and shall be designed to be as the result of a crash which may have prevented the driver from initiating the emergency request personally.

Data Flows: All input data flows are unsolicited and all output flows are solicited, with the exception of the data read from the store of vehicle identity.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow of data from sensors indicating that the vehicle has been involved in a crash;
(b) when the input in (a) is received, get the cargo state by sending a request to the monitoring process and extracting the data from the reply;
(c) if no data is received in (b) within a specified time out period, assume that there is no cargo and build the message for output to the communications process using the vehicle location and identity data inputs listed above;
(d) when all the data has been assembled, output the data flow to the communications process;
(e) the processing of the sensor input and generation of the output shall be completed within a time frame consistent with the safe operation of vehicle control systems with human interfaces regardless of the amount of processing needed to produce the output data flow to the communications process.

User Service Requirements:

USR = 5.0;
USR = 5.1;
USR = 5.1.1;
USR = 5.1.1.4;
USR = 5.1.2;
USR = 5.1.2.1;
USR = 5.1.2.1.1;
USR = 5.1.2.1.2;
USR = 5.1.2.2;
USR = 5.1.2.2(a);

Output Flow Dynamics Assumptions:

cargo_data_request = 1/(60*60*24*7*52);
vehicle_emergency_request = 1/(60*60*24*7*52);

3.4 Enhance Driver's Vision

Input Flows

From_Roadway

Output Flows

vision_data

Description:

Overview: This process shall be responsible for providing data from which a continuously updated display showing an enhanced version of the driver's vision. The process shall produce the data for this display using inputs to sensors mounted on the vehicle. It shall operate at all times and shall send its output to another process for integration with other messages for the driver.

Functional Requirements: This process shall meet the following functional requirements:
(a) continually receive data from the roadway through in-vehicle sensors;
(b) process the data received in (a) to produce an enhanced picture of the driver's view from the vehicle, in the forward, reverse or sideways directions;
(c) the enhancement provided in the data must both improve the field of view, i.e. the distance of the farthest point which can be seen and the width of view, and the clarity with which all objects within that distance can be seen;
(d) the resulting data must be continuously sent in the vision data output to another process for combination with other driver output data;
(e) complete a full scan of the input and generate the output in a time frame consistent with the safe operation of vehicle control systems and human interfaces regardless of the size of the input (field of view) to be scanned.

User Service Requirements:

USR = 6.0;
USR = 6.4;
USR = 6.4.0;
USR = 6.4.1;

Output Flow Dynamics Assumptions:

vision_data = 1;

4.1.1 Process Transit Vehicle Sensor Trip Data

Input Flows

ftv_vehicle_trip_data

Output Flows

transit_vehicle_arrival_times
transit_vehicle_collected_trip_data
transit_vehicle_on_board_data

Description:

Overview: This process shall collect and process data available to sensors on-board transit vehicles. This data shall be sent by this process to other processes on-board the transit vehicle and elsewhere in the Manage Traffic function for use in determining vehicle schedule deviations and for storage as operations data.

Data Flows: The input data flow is unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the outputs identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.1;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.1;

Output Flow Dynamics Assumptions:

transit_vehicle_collected_trip_data = 1/(60)*ITS_TRANSIT_VEHS;
transit_vehicle_on_board_data = 1;
transit_vehicle_arrival_times = 1/(60)*ITS_TRANSIT_VEHS;

4.1.2.1 Determine Transit Vehicle Deviation and ETA

Input Flows

transit_services_for_eta
transit_vehicle_arrival_times
transit_vehicle_location_for_eta

Output Flows

transit_services_for_eta_request
transit_vehicle_deviation_data
transit_vehicle_deviations
transit_vehicle_deviations_from_schedule
transit_vehicle_eta
transit_vehicle_eta_for_advisory
transit_vehicle_schedule_deviation

Description:

Overview: This process shall determine the schedule deviation and estimated times of arrival (ETA) at transit stops of a transit vehicle. The data shall be sent by this process to other processes in the Manage Transit function for use in calculating corrective instructions for output to the transit vehicle drivers, for use in calculation of a much wider return to schedule strategy where more than one vehicle and/or service is involved, and for storage as transit vehicle operational data. This process shall also send the data to the transit driver interface process, so that the driver is aware of the actual schedule deviation. This output shall be set to zero (no deviation) when that condition occurs, even when it has followed a period of deviation from schedule.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above using the methods described below and with the exceptions noted below;
(c) generate the schedule deviation using the current service details (routes and schedules) and the vehicle's current location using interpolation or some other algorithmic method;
(d) use similar methods to generate the transit vehicle's estimated time of arrival at the next transit stop for output to the stop as the vehicle approaches;
(e) only if the deviation is small and in an urban area the process shall send the data to the process that generates corrective instructions;
(f) if the deviation is large or not in an urban area, the process shall not produce the output identified in (e) above and shall instead just send the data to the process that manages transit vehicle deviations;
(g) the process shall generate outputs even when the deviation is zero.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.1.2.1.1;
USR = 2.1.1.2.4;

Output Flow Dynamics Assumptions:

transit_services_for_eta_request = 1/(60)*ITS_TRANSIT_VEHS;
transit_vehicle_deviations = 1/(60)*TRANSIT_DEVS;

4.1.2.2 Determine Transit Vehicle Corrective Instructions

Input Flows

approved_corrective_plan
transit_services_for_corrections
transit_vehicle_deviations

Output Flows

transit_vehicle_arrival_conditions
transit_vehicle_corrective_instructions
transit_vehicle_preemption_request

Description:

Overview: This process shall generate outputs that enable a transit vehicle schedule deviation to be corrected. The process shall derive its outputs from data received from another process in the Manage Traffic function. The outputs produced by the process shall consist of corrective instructions for output to the transit vehicle driver by a process on-board the vehicle, and preemption requests for traffic signal controllers at road and freeway intersections. The process shall only produce this output when another process has determined that deviation is small, or the transit vehicle is operating in an urban area. In all other conditions, the process shall provide an output that shows that there are no corrective instructions.

Data Flows: The input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input flow of transit vehicle deviations is received, generate the outputs identified above using any appropriate algorithms for determining the corrective instruction data for the transit vehicle driver.

transit_vehicle_deviation_data = 1/(60)*TRANSIT_DEVS;
transit_vehicle_deviations_from_schedule = 1/(60)*TRANSIT_DEVS;

User Service Requirements: transit_vehicle_eta = 1/(60)*TRANSIT_DEVS;

transit_vehicle_schedule_deviation = 1/(60)*TRANSIT_DEVS; USR = 2.0;
transit_vehicle_eta_for_advisory = 1/(60)*TRANSIT_DEVS; USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.1.2.1.4;
USR = 2.1.1.2.2;
USR = 2.1.1.2.4;

Output Flow Dynamics Assumptions:

transit_vehicle_arrival_conditions = 1/(60)*TRANSIT_DEVS;
transit_vehicle_preemption_request = 1/(60)*TRANSIT_DEVS;
transit_vehicle_corrective_instructions = 1/(60)*TRANSIT_DEVS;

4.1.2.3 Provide Transit Vehicle Driver Interface

Input Flows

transit_vehicle_corrective_instructions
transit_vehicle_deviation_data

Output Flows

ttd_corrective_instructions
ttd_transit_vehicle_schedule_deviations

Description:

Overview: This process shall provide a schedule correction interface for the transit driver in the transit vehicle. The interface shall provide data to the driver about how far the vehicle is from its schedule and what corrective action the driver must take. The data shall be received by the process from other processes in the Manage Traffic function. The output delivered by the process shall be available in audio or visual form in such way that while alerting the driver to the information it contains, it shall in no way impair the driver's ability to operate the vehicle in a manner that is both safe to its passengers and to other vehicles on the roads and freeways. The process shall maintain the output until new data is received from the other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_vehicle_corrective_instructions';
(b) 'transit_vehicle_deviation_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ttd-corrective_instructions';
(b) 'ttd-transit_vehicle_schedule_deviations'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) the output shall be presented in such a way that it provides the necessary information without jeopardizing the driver's ability to operate the vehicle safely, both for its passengers and for other vehicles on the roads and freeways;
(c) the output shall be maintained until fresh data is received through the unsolicited input flows listed above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.1.2.1.3;
USR = 2.1.1.2.1.4;
USR = 2.1.1.2.1.4(a);
USR = 2.1.1.2.1.4(b);

Output Flow Dynamics Assumptions:

ttd-corrective_instructions = transit_vehicle_corrective_instructions;
ttd-transit_vehicle_schedule_deviations = transit_vehicle_deviation_data;

4.1.2.4 Provide Transit Vehicle Correction Data Output Interface

Input Flows

traffic_data_for_transit
transit_vehicle_arrival_conditions

Output Flows

tmtsp_transit_arrival_changes

Description:

Overview: This process shall provide the interface through which multimodal transportation service providers are informed of a transit vehicle schedule deviation. The output delivered by the process results from input received from another process in the Manage Transit function, and shall relate to the deviation of an individual transit vehicle. The process shall provide the output in a form that enables adjustments to be made to any connecting services being provided by the multimodal supplier so that transit users are not inconvenienced by the deviation of a transit vehicle on one service.
A zero (or null) output shall be provided when no deviations are present.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_vehicle_arrival_conditions';
(b) 'traffic_data_for_transit'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmtsp-transit_arrival_changes'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) provide the solicited output flow listed above in a form that will enable the multimodal service provider to take any remedial action to the service it provides so that transit users suffer the minimum of inconvenience as a result of a late running service;
(c) if no input is received, or it shows that there are no deviations, set the output to zero (or null) to reflect this condition.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.2;
USR = 2.1.2.2.4;

Output Flow Dynamics Assumptions:

tmtsp-transit_arrival_changes = transit_vehicle_arrival_conditions;

4.1.2.5 Request Transit Vehicle Preemptions

Input Flows

transit_vehicle_preemption_request

Output Flows

transit_vehicle_roadway_preemptions

Description:

Overview: This process shall provide the interface through which requests for preemption can be output from a transit vehicle. The output shall be received by the process as a result of data sent from another process in the Manage Transit function. The process shall provide the output in a form that can be used by the controllers at intersections, pedestrian crossings and multimodal crossings on the roads (surface streets) and freeway network served by the Manage Traffic function to provide priority of the transit vehicle. If no data is received from the other process, or it shows that no preemption is needed, the process shall produce no output.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_vehicle_preemption_request'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_vehicle_roadway_preemptions'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the flow listed above is received, produce the solicited output flow listed above in a form that can be used by roadside intersection controllers to give priority to the transit vehicle;
(c) if no input flow is received, or it indicates that preemption is not required, produce no output data flow.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.1.2.1.4;
USR = 2.1.1.2.2;
USR = 2.1.1.2.3;

Output Flow Dynamics Assumptions:

transit_vehicle_roadway_preemptions = transit_vehicle_preemption_request;

4.1.3 Provide Transit Vehicle Location Data

Input Flows

transit_vehicle_on_board_data
vehicle_location_for_transit

Output Flows

transit_vehicle_location
transit_vehicle_location_for_deviation
transit_vehicle_location_for_eta
transit_vehicle_location_for_store

Description:

Overview: This process shall provide the transit vehicle's current location with a high degree of accuracy. The location shall be computed by this process from data sent by other processes that provides basic vehicle location and on-board vehicle conditions, such as proximity to transit stop, vehicle doors opened or closed, etc. The data shall be output continuously by the process and sent to other processes for their use and for storage.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when any of the inputs are received, generate the outputs identified above by combining the new data with the last output values;
(c) the calculation of the new location shall use the basic location data and refine it by use of data from on-board the vehicle, e.g. proximity of transit stop, vehicle doors open, etc.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.1;
USR = 2.1.1.1(d);
USR = 2.1.1.1(f);
USR = 2.1.2;
USR = 2.1.2.2;
USR = 2.1.2.2.1;
USR = 2.1.2.2.1(a);
USR = 2.1.2.2.1(b);

Output Flow Dynamics Assumptions:

transit_vehicle_location = 1*ITS_TRANSIT_VEHS;
transit_vehicle_location_for_deviation = 1*ITS_TRANSIT_VEHS;
transit_vehicle_location_for_eta = 1*ITS_TRANSIT_VEHS;
transit_vehicle_location_for_store = 1*ITS_TRANSIT_VEHS;

4.1.4 Manage Transit Vehicle Deviations

Input Flows

ftfm_approved_corrections
prediction_data
transit_highway_priority_given
transit_ramp_priority_given
transit_road_priority_given
transit_services_for_scenarios
transit_vehicle_deviations_from_schedule
transit_vehicle_location_for_deviation

Output Flows

approved_corrective_plan
transit_highway_overall_priority
transit_ramp_overall_priority
transit_road_overall_priority
transit_vehicle_arrival_deviations
transit_vehicle_deviation_update
ttfm_proposed_corrections

Description:

Overview: This process shall manage large deviations of individual transit vehicles, deviations in rural areas, and deviations of large numbers of vehicles. The process shall generate the necessary corrective actions which may involve more than the vehicles concerned and more far reaching action, such as, the introduction of extra vehicles, wide area signal preemption by the Manage Traffic function, the premature termination of some services, etc. All corrective actions generated by this process shall be subject to the approval of the transit fleet manager before being implemented. Confirmation that the requested overall priority has been given by the Manage Traffic function shall be received by the process.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input of deviation from schedule is received use the current transit services and the forecast of traffic conditions provided by the data in the flow of predictive data received from the Manage Traffic function as input to algorithms that can produce a return to service strategy;
(c) when the process in (b) is complete, send the strategy to the transit fleet manager for approval;
(d) when the approval in (c) is received, send the remaining output flows, and await confirmation of the implementation of the requested priority by the Manage Traffic function;
(e) if the confirmation on (d) is not given, then re-implement (b) through (d) using the fact that priority requests to the Manage Traffic function are not available.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.1.2.1.4;
USR = 2.1.1.2.2;
USR = 2.1.1.2.3;
USR = 2.1.1.2.4;

Output Flow Dynamics Assumptions:

approved_corrective_plan = 1/(60)*TRANSIT_DEVS;
transit_vehicle_arrival_deviations = 1/(60)*TRANSIT_DEVS;
transit_highway_overall_priority = 1/(60)*TRANSIT_DEVS;
transit_road_overall_priority = 1/(60)*TRANSIT_DEVS;
transit_ramp_overall_priority = 1/(60)*TRANSIT_DEVS;
transit_vehicle_deviation_update = (1/(60)*TRANSIT_DEVS)/TRANSIT_STOPS;
ttfm-proposed_corrections = 1/(60)*TRANSIT_DEVS;

4.1.5 Provide Transit Vehicle Status Information

Input Flows

ftfm_request_transit_vehicle_data
transit_conditions_demand_request
transit_vehicle_information

Output Flows

transit_information_request
transit_probe_data
transit_running_data_for_demand
transit_vehicle_arrival_time
transit_vehicle_data
transit_vehicle_data_for_archive
transit_vehicle_status
ttfm_transit_vehicle_data

Description:

Overview: This process shall provide transit vehicle operational data to processes within the Manage Transit function, and on request to the transit fleet manager and the Manage Travel Demand facility in the Manage Traffic function. This process shall also provide transit probe and AVL information to the Manage Traffic function. Transit probe information can be provided by fixed route, flexibly routed, and paratransit services. The data shall be obtained by this process from another process that manages a store of transit vehicle operating data.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) at regular periodic intervals, send the request for transit information;
(c) when the data in (b) is received, send out the data flows listed above that do not have any corresponding data request flow;
(d) when any other input listed above is received, output the flow requesting transit vehicle information and send the response back to the originating process.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.1.2.1.2;
USR = 2.1.1.2.1.3;

Output Flow Dynamics Assumptions:

transit_information_request = 12/HOUR;
transit_running_data_for_demand = transit_conditions_demand_request;
transit_vehicle_arrival_time = (1/HOUR)*TRANSIT_STOPS;
transit_vehicle_data = 12/HOUR;
transit_vehicle_status = 12/HOUR;
ttfm-transit_vehicle_data = ftfm-request_transit_vehicle_data;
transit_probe_data = 12/(60*60)*TRANSIT_PROBE_VEHS;
transit_vehicle_data_for_archive = 1/(60*60);

4.1.6 Manage Transit Vehicle Operations Data

Input Flows

fm_transit_schedule_deviations_request
fws_current_weather
fws_predicted_weather
traffic_data_for_transit
transit_information_request
transit_vehicle_collected_maintenance_data
transit_vehicle_collected_trip_data
transit_vehicle_deviation_update
transit_vehicle_deviations_details_request
transit_vehicle_eta
transit_vehicle_location_for_store
transit_vehicle_operating_data
transit_vehicle_schedule_deviation

Output Flows

tm_transit_schedule_deviations_to_media
transit_deviation_data_received
transit_vehicle_advisory_eta
transit_vehicle_collected_maintenance_data_request
transit_vehicle_deviations_details
transit_vehicle_information
transit_vehicle_operating_data
transit_vehicle_user_data

Description:

Overview: This process shall manage the store of transit vehicle operating data. When any new data is received from another process, this process shall load it into the data store. This process shall also retrieve selected data on request from other processes in the Manage Transit function.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from and written to a data store:
(a) 'transit_vehicle_operating_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs that are not data requests are received, load the data that they contain into the data store using the flow identified above and generate the update and transit vehicle information output flows identified above;
(c) periodically generate requests for data collected by sensors on-board the vehicle and store the solicited data when received
(d) when inputs that are data requests are received, retrieve the data and only send it to the source of the request;
(e) the transit user data flow that is sent to each transit stop shall be sent as each transit vehicle approaches a stop to provide information for output to transit users at the stop;
(f) be responsible for the management of the data in the store of transit vehicle operational data.

User Service Requirements:

USR = 2.0;
USR = 2.1;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.1;
USR = 2.1.1.1(a);
USR = 2.1.1.1(b);
USR = 2.1.1.1(e);

Output Flow Dynamics Assumptions:

transit_deviation_data_received = transit_vehicle_deviation_update;
transit_vehicle_collected_maintenance_data_request = 1/(60)*ITS_TRANSIT_VEHS;
transit_vehicle_deviations_details = transit_vehicle_deviations_details_request;
transit_vehicle_operating_data = 1/(60)*ITS_TRANSIT_VEHS+12/(60*60);
transit_vehicle_information = 12/(60*60);
transit_vehicle_user_data = TRANSIT_STOPS*(2/60);
transit_vehicle_advisory_eta = 1/(60)*TRANSIT_DEVS;
tm-transit_schedule_deviations_to_media = transit_vehicle_schedule_deviation;

4.1.7 Provide Transit Vehicle Deviation Data Output Interface

Input Flows

transit_vehicle_arrival_deviations

Output Flows

tmtsp_transit_arrival_deviations

Description:

Overview: This process shall provide the interface through which multimodal transportation service providers are informed of transit vehicle schedule deviations. The output delivered by the process shall result from input received from another process in the Manage Transit function, and shall relate to the deviation of a number of transit vehicles such that the disruption will affect several services, possibly on different routes. The process shall provide the output in a form that enables adjustments to be made to any connecting services being provided by the multimodal supplier so that transit users are not inconvenienced by the deviations. A zero (or null) output shall be provided when no deviations are present.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_vehicle_arrival_deviations'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmtsp-transit_arrival_deviations'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) provide the solicited output flow listed above in a form that will enable the multimodal service provide to take any remedial action to the service it provides so that transit users suffer the minimum of inconvenience as a result of a late running service;
(c) if no input is received, or it shows that there are no deviations, set the output to zero (or null) to reflect this condition.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.2;
USR = 2.1.2.2;
USR = 2.1.2.2.4;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.2;
USR = 2.2.1.1.3;
USR = 2.2.1.1.4;

Output Flow Dynamics Assumptions:

tmtsp-transit_arrival_deviations = transit_vehicle_arrival_deviations;

4.1.8 Provide Transit Operations Data Distribution Interface

Input Flows

fm_transit_vehicle_deviations_request
transit_conditions_advisories_request
transit_conditions_guidance_request
transit_deviation_data_received
transit_deviation_kiosk_request
transit_deviations_personal_request
transit_vehicle_deviations_details
traveler_transit_profile

Output Flows

tm_transit_vehicle_deviations
transit_deviation_kiosk_request_for_archive
transit_deviations_for_broadcast_to_kiosks
transit_deviations_for_broadcast_to_personal_devices
transit_deviations_for_kiosks
transit_deviations_for_personal_devices
transit_deviations_personal_request_for_archive
transit_running_data_for_advisory_output
transit_running_data_for_guidance
transit_vehicle_deviations_details_request

Description:

Overview: This process shall provide customized sets of transit vehicle schedule deviations to travelers, the traveler information data archive, and to the media. The process shall only provide data to the media and data archive when prompted by the arrival of new deviation data in the transit_vehicle_operational_data store, which is maintained by another process in the Manage Transit function. The outputs shall be made available following a direct request from the other ITS function, or as part of a subscription process relating to a traveler's transit profile. The process shall obtain the required data from the process that manages the store of transit vehicle operating data. The process shall send kiosk and personal transit deviation requests to the archival process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_conditions_advisories_request';
(b) 'transit_conditions_guidance_request';
(c) 'transit_deviation_data_received';
(d) 'transit_deviation_kiosk_request';
(e) 'transit_deviations_personal_request';
(f) 'fws-predicted_weather';
(g) 'fws-current_weather'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'transit_vehicle_deviations_details'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tm-transit_schedule_variations';
(b) 'transit_running_data_for_advisory_output';
(c) 'transit_running_data_for_guidance';
(d) 'transit_vehicle_deviations_details_request';
(e) 'transit_deviations_for_kiosks';
(f) 'transit_deviations_for_personal_devices';
(g) 'transit_deviation_kiosk_request_for_archive';
(h) 'transit_deviations_personal_request_for_archive';
(i) 'transit_deviations_for_broadcast_to_kiosks';
(j) 'transit_deviations_for_broadcast_to_personal_devices'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) generate the transit deviations details message and output a copy of the deviations requests to the archive when any of the inputs are received and return the data received in the solicited input flow to the source(s) of the input(s);
(c) only provide output to the media when the unsolicited input, transit (vehicle) deviation data received, is received and the data provided by the solicited input flow has also been received.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.2;
USR = 2.1.2.2;
USR = 2.1.2.2.1;
USR = 2.1.2.2.5;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(g);

Output Flow Dynamics Assumptions:

tm-transit_schedule_variations = transit_deviation_data_received;
transit_running_data_for_advisory_output = transit_conditions_advisories_request;
transit_running_data_for_guidance = transit_conditions_guidance_request;
transit_vehicle_deviations_details_request = transit_deviation_data_received;
transit_deviations_for_kiosks = transit_deviation_kiosk_request;
transit_deviations_for_personal_devices = transit_deviations_personal_request;
transit_deviations_for_broadcast_to_kiosks = transit_deviation_kiosk_request;
transit_deviations_for_broadcast_to_personal_devices = transit_deviations_personal_request;
tm-transit_vehicle_deviations = transit_vehicle_deviations_details;
transit_deviation_kiosk_request_for_archive = transit_deviation_kiosk_request;
transit_deviations_personal_request_for_archive = transit_deviations_personal_request_for_archive;

4.1.9 Process Transit Vehicle Sensor Maintenance Data

Input Flows

ftv_vehicle_maintenance_data
transit_vehicle_collected_maintenance_data_request

Output Flows

transit_vehicle_collected_maintenance_data

Description:

Overview: This process shall collect and process vehicle maintenance data available to sensors on-board transit vehicles. When processed, the data shall be sent by this process on request to another process in the Manage Transit function for storage as transit vehicle operating data so that it can subsequently be used for work on future vehicle maintenance.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when the vehicle maintenance data flow is received, process it and if required translate it into a digital form;
(c) when the request for transit vehicle collected maintenance data is received, generate the output data flow identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.1;
USR = 2.1.1.1(c);
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.1;

Output Flow Dynamics Assumptions:

transit_vehicle_collected_maintenance_data = transit_vehicle_collected_maintenance_data_request;

4.2.1.1 Process Demand Responsive Transit Trip Request

Input Flows

paratransit_schedule
paratransit_service_confirmation
paratransit_service_data
paratransit_trip_request

Output Flows

paratransit_personal_schedule
paratransit_request
paratransit_requested_services
paratransit_service_data
paratransit_service_data_for_archive

Description:

Overview: This process shall provide the interface through which processes in the Provide Driver and Traveler Service function can gain access to the Provide Demand Responsive Transit Service facility. The process shall enable the interface to support the receipt of trip requests, their transfer to another process for the actual demand responsive schedule generation, the output of the proposed schedule and their (possible) subsequent confirmation. The process shall store the input and schedule data relating to each request until such time as the request is confirmed or the data in the request is no longer valid, e.g. the time(s) used in the proposed schedule has(ve) passed. The confirmation of a particular schedule shall be sent by the process to another process that will enable the schedule to be implemented.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from and written to the store of request data:
(a) 'paratransit_service_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the flow in (a) is a new schedule request, generate the corresponding output identified above and send it to the schedule generation process;
(c) as a result of (b) continuously monitor for the receipt of the flow listed above containing details of the proposed schedule;
(d) when the flow in (c) is received, load the data into the store of paratransit service data, including an identity number with it;
(e) when (d) is successfully complete, generate the output listed above that contains the details of the proposed service for use by the requesting process;
(f) when the flow in (a) is a schedule confirmation, read the schedule data corresponding to the identity number and generate the output to the schedule confirmation process identified above;
(g) manage the data in the store of trip request data.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.1;
USR = 2.3.1.1;
USR = 2.3.1.2;
USR = 2.3.2;
USR = 2.3.2.7;

Output Flow Dynamics Assumptions:

paratransit_personal_schedule = 5/(60*60)*ITS_PTRANSIT_TRAVS;
paratransit_requested_services = 1/(60*60)*ITS_PTRANSIT_TRAVS;
paratransit_service_details = 5/(60*60)*ITS_PTRANSIT_TRAVS;
paratransit_request = paratransit_trip_request;
paratransit_service_data_for_archive = paratransit_trip_request;

4.2.1.2 Compute Demand Responsive Transit Vehicle Availability

Input Flows

paratransit_transit_vehicle_availability
transit_vehicle_location

Output Flows

paratransit_available_vehicles

Description:

Overview: This process shall provide the facility for the calculation of the location and availability of transit vehicles for use in demand responsive transit operations. The process shall base its calculation on the vehicle's current location and on the output from a process that determines vehicle availability from data input to sensors. The output shall be loaded by the process into a store for use by another process.

Data Flows: All input data flows are unsolicited and the output flow shall be sent to the store of available transit vehicles using the flow 'paratransit_available_vehicles'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the output identified above;
(c) manage the data in the store of transit vehicle availability.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.2;
USR = 2.3.2.6;
USR = 2.3.2.7;

Output Flow Dynamics Assumptions:

paratransit_available_vehicles = 1/(60)*ITS_PTRANSIT_VEHS;

4.2.1.3 Generate Demand Responsive Transit Schedule and Routes

Input Flows

paratransit_available_vehicles
paratransit_request
paratransit_services
traffic_data_for_transit
transit_services_for_demand_response

Output Flows

paratransit_schedule
paratransit_services
transit_services_demand_response_request

Description:

Overview: This process shall provide dynamic routing and scheduling of transit vehicles so that a demand responsive transit service can be provided. The generation of the specific route and schedule by the process shall be initiated by a request from the management process. The choice of route and schedule produced by the process shall depend on what other demand responsive transit schedules have been planned, the availability and location of vehicles, and the relevance of any regular transit routes and schedules. The process shall send its output to another process for output to the requesting process, and shall also load it into a data store for use if the schedule is later confirmed.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'paratransit_services', which contains data requested from and written to a data store;
(b) 'paratransit_available_vehicles', which also contains data requested from a data store;
(c) 'transit_services_for_demand_response', which is received as a result of output being sent to another process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above using an appropriate routes and schedule generation algorithm;
(c) manage of the data in the paratransit services data store, including generated routes and schedules;
(d) use appropriate database mechanism(s) to retrieve data from the store of available transit vehicles identified above.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.2;
USR = 2.3.2.1;
USR = 2.3.2.10;
USR = 2.3.2.2;
USR = 2.3.2.3;
USR = 2.3.2.4;
USR = 2.3.2.5;
USR = 2.3.2.6;
USR = 2.3.2.7;
USR = 2.3.2.8;
USR = 2.3.2.9;
USR = 2.3.4;
USR = 2.3.4.2;

Output Flow Dynamics Assumptions:

paratransit_schedule = 1/(60*60)*ITS_PTRANSIT_TRAVS;
paratransit_services = 1/(60*60)*ITS_PTRANSIT_TRAVS;
transit_services_demand_response_request = 1/(60*60)*ITS_PTRANSIT_TRAVS;

4.2.1.4 Confirm Demand Responsive Transit Schedule and Route

Input Flows

paratransit_requested_services
paratransit_services

Output Flows

paratransit_service_output
paratransit_services_for_transit_drivers
paratransit_transit_driver_instructions
ttfm_paratransit_service

Description:

Overview: This process shall provide output when a demand responsive transit schedule is confirmed. The outputs shall contain details of the schedule and shall be sent to the transit fleet manager and to processes that provide interfaces to the transit driver, a store of data used by the regular transit routes and schedule generation processes, and the transit driver schedule generation processes. The process shall obtain the data for the outputs from the store of data provided by the schedule generation process.

Data Flows: The input data flow is unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'paratransit_services'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the outputs identified above;
(c) manage the data in the store of paratransit services.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.1;
USR = 2.3.1.3;
USR = 2.3.2;
USR = 2.3.2.2;
USR = 2.3.2.3;
USR = 2.3.4;
USR = 2.3.4.3;

Output Flow Dynamics Assumptions:

paratransit_services_for_transit_drivers = paratransit_service_confirmation;
paratransit_service_output = paratransit_service_confirmation;
paratransit_transit_driver_instructions = paratransit_service_confirmation;
ttfm-paratransit_service = paratransit_service_confirmation;

4.2.1.5 Process Demand Responsive Transit Vehicle Availability Data

Input Flows

ftv_availability

Output Flows

paratransit_transit_vehicle_availability

Description:

Overview: This process shall manage data input to sensor(s) on board a transit vehicle. Data including the vehicle's availability for use in demand responsive transit services shall be provided by this process to other processes within the Manage Transit function.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'ftv-availability'.

Solicited Output Processing: This process shall provide the following output flow as a result of the above inputs being received:
(a) 'paratransit_transit_vehicle_availability'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow shown above;
(b) analyze the input flow and if required, transform it into digital data for use by other processes;
(c) send the data in the solicited output flow shown above to the process that will combine it with vehicle location.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.3;
USR = 2.3.3.1;
USR = 2.3.3.1(a);
USR = 2.3.3.1(b);
USR = 2.3.3.1(c);
USR = 2.3.3.2;
USR = 2.3.3.2(a);
USR = 2.3.3.2(b);
USR = 2.3.3.3;

Output Flow Dynamics Assumptions:

paratransit_transit_vehicle_availability = ftv-availability;

4.2.1.6 Provide Demand Responsive Transit Driver Interface

Input Flows

paratransit_transit_driver_instructions

Output Flows

ttd_paratransit_information

Description:

Overview: This process shall provide the interface through which a transit driver will be sent instructions about the demand responsive transit schedule that has been confirmed. The process shall send the data in a format that will enable the driver to implement the schedule. The output provided by the process shall be available in audio or visual form in such a way that while alerting the driver to the information it contains, it shall in no way impair the driver's ability to operate the vehicle in a manner that is both safe to its passengers, and to other vehicles on the roads and freeways. The input and output forms shall also include those that are suitable for travelers with physical disabilities.

(a) 'paratransit_transit_driver_instructions'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ttd-paratransit_information'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) produce the output to the transit driver in such a way that it does not jeopardize the driver's safe operation of the transit vehicle, but conveys the required information in an easily understandable form;
(c) maintain the output for as long as the schedule is current, i.e. until the time of the last activity or the last arrival time has past.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.5;
USR = 2.3.5.1;
USR = 2.3.5.2;
USR = 2.3.5.2(a);
USR = 2.3.5.2(b);
USR = 2.3.5.3;
USR = 2.3.5.4;
USR = 2.3.5.4(a);
USR = 2.3.5.4(b);

Output Flow Dynamics Assumptions:

ttd-paratransit_information = paratransit_transit_driver_instructions;

4.2.2 Provide Transit Plans Store Interface

Input Flows

paratransit_service_output
transit_plans
transit_routes_request
transit_routes_updates
transit_schedule_request
transit_schedule_updates
transit_services_demand_response_request

Output Flows

transit_plans
transit_routes_current_data
transit_schedule_current_data
transit_services_for_demand_response

Description:

Overview: This process shall provide the interface to the store of current regular transit plans, i.e., routes and schedules and demand responsive transit schedules. The process shall enable the store to be used by the Demand Responsive Transit facility as a source of data about regular transit services when it is generating its schedules. The demand responsive transit schedule data shall be accessible as input to the regular transit route and schedule generation processes.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from and written to the store of transit plans:
(a) 'transit_plans'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) manage the data in the store of transit plans data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.1;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.4;

Output Flow Dynamics Assumptions:

transit_plans = transit_routes_updates + transit_schedule_updates;
transit_routes_current_data = transit_routes_request;
transit_services_for_demand_response = transit_services_demand_response_request;
transit_schedule_current_data = transit_schedule_request;

4.2.3.1 Generate Transit Routes

Input Flows

map_data_for_transit
transit_operational_data
transit_routes_current_data
transit_service_planning_parameters
update_routes

Output Flows

transit_routes_data
transit_routes_request
transit_routes_updates

Description:

Overview: This process shall generate new transit routes. The process shall use parameters set up by the transit fleet manager, operational data for the current routes and schedules, plus the current routes and digitized map data, as sources of input from which the new routes are generated. The process shall also use the requested input data containing the demand responsive transit routes and schedules. The generation of new routes by the process shall be initiated as a result of data received from the transit fleet manager interface process, with the output being sent to other processes for storage. The output data produced by the process shall include sufficient data for a specialist map data provider to generate maps showing transit routes and stops, either as separate data or as part of the general digitized map data provided to other ITS functions.

Data Flows: The input data flow for updating routes and services is unsolicited and all other input and output flows are then solicited as a result of its receipt. The following data flows are received as a result of requests for data from stores:
(a) 'map_data_for_transit';
(b) 'transit_operational_data';
(c) 'transit_service_planning_parameters'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow requesting update of the routes and services from the transit fleet manager and shown in the list above;
(b) when this input is received, initiate the generation process, reading in all the required data from stores, or requesting it from the store interface process;
(c) use the data in (b) to produce the output of new transit routes data using an appropriate route generation algorithm;
(d) use appropriate database mechanism(s) to retrieve data from the stores identified by the flows shown above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;

Output Flow Dynamics Assumptions:

transit_routes_data = update_routes;
transit_routes_request = update_routes;
transit_routes_updates = update_routes;

4.2.3.2 Generate Schedules

Input Flows

parking_lot_transit_request
transit_operational_data
transit_schedule_current_data
transit_service_planning_parameters
update_schedules

Output Flows

parking_lot_transit_response
transit_schedule_data
transit_schedule_request
transit_schedule_updates

Description:

Overview: This process shall generate new transit schedules for use by the regular transit operation. The process shall use parameters set up by the transit fleet manager, operational data for the current routes and schedules, plus the current routes and schedules themselves, as sources of input from which the new schedules are generated. The process shall also use the data containing the demand responsive

transit routes and schedules to generate the new schedules. The generation of new schedules by the process shall be initiated as a result of data received from the transit fleet manager interface process or a request for services to a parking lot. The process shall send its output to another process for storage.

Data Flows: The input data flow for updating routes and services is unsolicited and all other input and output flows are then solicited as a result of its receipt. The following data flows are as a result of requests for data from stores:
(a) 'transit_operational_data';
(b) 'transit_service_planning_parameters'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow requesting update of the schedules shown in the list above;
(b) when this input is received, initiate the generation process, reading in all the required data and produce the output of the new transit schedules data using an appropriate schedule generation algorithm;
(c) use appropriate database mechanism(s) to retrieve data from the stores identified by the flows shown above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.1;
USR = 2.1.2.2;
USR = 2.1.2.2.2;
USR = 2.1.2.2.3;
USR = 2.1.2.2.5;
USR = 2.1.2.2;
USR = 2.1.2.2.1;
USR = 2.1.2.2.2;

Output Flow Dynamics Assumptions:

parking_lot_transit_response = parking_lot_transit_request;
transit_schedule_data = 1/(60*60*24*7);
transit_schedule_request = 1/(60*60*24*7);
transit_schedule_updates = 1/(60*60*24*7);

4.2.3.3 Produce Transit Service Data for External Use

Input Flows

transit_service_external_data
transit_services_advisories_request
transit_services_demand_request
transit_services_guidance_request
transit_services_kiosk_request
transit_services_personal_request
transit_services_travelers_request

Output Flows

request_transit_service_external_data
tmtsp_transit_service_data
transit_services_for_advisory_data
transit_services_for_demand
transit_services_for_deployment
transit_services_for_guidance
transit_services_for_kiosks
transit_services_for_personal_devices
transit_services_for_travelers

Description:

Overview: This process shall obtain transit routes and services data and distribute it to ITS functions that are outside the transit center. The process shall run when a request for data is received from an external source, or when fresh data is received. In the latter case, the data shall only be sent by the process to the multimodal transportation service provider. For data requests that include an origin and a destination, the process shall only provide details of the transit service(s) that link the two points. The details shall only cover those portion(s) of the service(s) that are needed to complete the requested trip and not full details of the services.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_services_advisories_request';
(b) 'transit_services_demand_request';
(c) 'transit_services_guidance_request';
(d) 'transit_services_kiosk_request';
(e) 'transit_services_travelers_request';
(f) 'transit_services_personal_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to another process:
(a) 'transit_service_external_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'request_transit_service_external_data';
(b) 'tmtsp-transit_service_data';
(c) 'transit_services_for_advisory_data';
(d) 'transit_services_for_demand';
(e) 'transit_services_for_deployment';
(f) 'transit_services_for_guidance';
(g) 'transit_services_for_kiosks';
(h) 'transit_services_for_travelers';
'transit_services_for_personal_devices'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited inputs shown above except the last are received, the process shall immediately generate the first solicited output shown above;
(c) when the solicited input flow is received as a result of (b) begin generation of the requested data, only including the details necessary to meet the request, i.e. all of the transit routes and schedules provided in response to every request;
(d) data shall only be sent to the source from which the data request originated;
(e) before output, the process shall put the data into a format that is easily read and interpreted by external processes and can also be read by travelers and transit users with the minimum of further processing;
(f) if the second unsolicited input is received, i.e. fresh service data is received without being

requested, the data shall only be sent to the multimodal transportation service provider using the second solicited output flow.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.2;
USR = 2.3.2.2;
USR = 2.3.2.3;

Output Flow Dynamics Assumptions:

tmtsp-transit_service_data = ftfm-initiate_service_updates;
request_transit_service_external_data = 12/(60*60);
transit_services_for_advisory_data = transit_services_advisories_request;
transit_services_for_demand = transit_services_demand_request;
transit_services_for_deployment = 1/DAY;
transit_services_for_guidance = transit_services_guidance_request;
transit_services_for_kiosks = transit_services_kiosk_request;
transit_services_for_travelers = transit_services_travelers_request;
transit_services_for_personal_devices = transit_services_personal_request;

4.2.3.4 Provide Transit Fleet Manager Interface for Services Generation

Input Flows

ftfm_initiate_service_updates
ftfm_planning_parameters
ftfm_planning_parameters_update_request
ftfm_transit_display_update_request
ftfm_transit_services_output_request
map_data_for_transit
transit_services_changes_request
transit_services_data_for_output

Output Flows

request_transit_map_update
request_transit_services_data_for_output
transit_service_planning_parameters
transit_services_changes_response
ttfm_parameters
ttfm_transit_services_output
update_routes
update_schedules

Description:

Overview: This process shall provide the interface through which the transit fleet manager controls the generation of new routes and schedules (transit services). The transit fleet manager shall be able to review and update the parameters used by the routes and schedules generation processes and to initiate these processes. This process shall also act as the interface through which the Manage Demand facility in the Manage Traffic function can request changes to the current routes and schedules in its efforts to adjust the modal split of travelers' trips in order to make the most efficient use of the road and highway network served by the local ITS functions. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data written to a data store:
(a) 'transit_service_planning_parameters'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the transit fleet manager listed above;
(b) when the inputs in (a) are received, generate the appropriate outputs identified above;
(c) continuously monitor for receipt of any input data flows that may be produced by the output flows generated in (b);
(d) use appropriate database mechanism(s) to write data to the store of transit service planning parameters, the flow identified above when input of new/updated parameters is received from the transit fleet manager;
(e) the process shall allow the schedule generation process to be initiated on its own, but shall always initiate that process if initiation of the routes generation process is requested. I.e. it shall not be possible to have old schedules applied to newly generated routes.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.1;
USR = 2.1.2.1.2;
USR = 2.1.2.2;
USR = 2.1.2.2.2;
USR = 2.1.2.2.4;
USR = 2.1.2.2.4(a);
USR = 2.1.2.2.4(b);

Output Flow Dynamics Assumptions:

transit_service_planning_parameters = 1/WEEK;
transit_services_changes_response = 4/HOUR;
ttfm-parameters = 1/DAY;
update_routes = 1/WEEK;
update_schedules = 1/WEEK;
request_transit_services_data_for_output = 4/HOUR;
ttfm-transit_services_output = 1/DAY;
request_transit_map_update = 1/WEEK;

4.2.3.5 Manage Transit Operational Data Store

Input Flows

ftfm_passenger_loading_updates
transit_operational_data
transit_roadside_passenger_data
transit_vehicle_availability
transit_vehicle_data
transit_vehicle_passenger_data

Output Flows

transit_operational_data
transit_operational_data_for_archive
ttfm_passenger_loading_error

Description:

Overview: This process shall collect transit operational data and load it into a data store for use by the routes and schedules generation processes. The data shall be provided to this process by other processes in the Manage Transit function and shall enable an accurate picture of how routes and schedules are currently operating in terms of the numbers of vehicles that are available, the numbers of passengers that they are carrying, and the numbers of passengers passing through each roadside facility (transit stop).

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_roadside_passenger_data';
(b) 'transit_vehicle_passenger_data';
(c) 'transit_vehicle_availability';
(d) 'transit_vehicle_data'.

Solicited Input Processing: This process shall receive the following input flow as a result of data being sent to the transit fleet manager terminator:
(a) 'ftfm-passenger_loading_updates'.

The remaining data flows are solicited output flows.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the inputs is received, write the data into the store of operational data using the output flow shown above;
(c) when writing the data to the store, rationalize the two counts of the numbers of passengers for each transit route segment, (one being reported by the fare collection process and the other by the transit vehicle monitoring process), reporting any differences to the transit fleet manager;
(d) periodically, read the data from the transit operational data store and send it to the archive function using the solicited flow shown above;
(e) manage the data in the store of transit operational data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.1;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.4;
USR = 2.3.0;
USR = 2.3.4;

Output Flow Dynamics Assumptions:

transit_operational_data = transit_vehicle_availability + transit_vehicle_data
+ transit_roadside_passenger_data
+ transit_vehicle_passenger_data;
transit_operational_data_for_archive = 1/(60*60*24);
ttfm-passenger_loading_error = 1/(60*60*24);

4.2.3.6 Produce Transit Service Data for Manage Transit Use

Input Flows

transit_service_internal_data
transit_services_for_eta_request

Output Flows

request_transit_service_internal_data
transit_services_for_advanced_fares
transit_services_for_corrections
transit_services_for_eta
transit_services_for_roadside_fares
transit_services_for_scenarios
transit_services_for_transit_drivers
transit_services_for_vehicle_fares

Description:

Overview: This process shall obtain transit routes and services data and distribute it internally to other processes in the Manage Transit function. The process shall only provide its outputs when fresh data is received from another process. If this does not happen for a long period of time (days), then the process shall initiate its own request for fresh data.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_service_internal_data';
(b) 'transit_services_for_eta_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to another process:
(a) 'transit_service_internal_data'.

Unsolicited Output Processing: This process shall provide the following output flows regardless of any inputs that are received:
(a) 'request_transit_service_internal_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_services_for_corrections';
(b) 'transit_services_for_eta';
(c) 'transit_services_for_advanced_fares';
(d) 'transit_services_for_vehicle_fares';
(e) 'transit_services_for_roadside_fares';
(f) 'transit_services_for_scenarios';
(g) 'transit_services_for_transit_drivers'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the unsolicited input shown above is received, the process shall immediately generate all of the solicited outputs shown above;
(c) if the first unsolicited input is not received periodically, then the process shall generate the unsolicited output shown above.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.2;
USR = 2.3.2.2;
USR = 2.3.2.3;

Output Flow Dynamics Assumptions:

request_transit_service_internal_data = 1/(60*60*24);
transit_services_for_corrections = ftfm-initiate_service_updates + 1/(60*60*24);
transit_services_for_eta = ftfm-initiate_service_updates + 1/(60*60*24);
transit_services_for_advanced_fares = ftfm-initiate_service_updates + 1/(60*60*24);
transit_services_for_roadside_fares = ftfm-initiate_service_updates + 1/(60*60*24);
transit_services_for_vehicle_fares = ftfm-initiate_service_updates + 1/(60*60*24);
transit_services_for_scenarios = ftfm-initiate_service_updates + 1/(60*60*24);
transit_services_for_transit_drivers = ftfm-initiate_service_updates + 1/(60*60*24);

4.2.3.7 Provide Interface for Other TRM Data

Input Flows

fotrm_transit_services
transit_services_for_other_TRM

Output Flows

other_TRM_service_data
totrm_transit_services

Description:

Overview: This process shall provide the interface through which transit routes and schedules can be exchanged with other transit centers (Other TRM). This data shall be output when data is received from another (local) process and shall enable coordination between services provided by adjacent transit operations, particularly where they serve the same geographic areas. The process shall also provide routes and schedules to the local process when the data is received from other transit centers.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fotrm-transit_services';
(b) 'transit_services_for_other_TRM'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'other_TRM_service_data':
(b) 'totrm-transit_services'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either of the unsolicited input flows is received, generate the corresponding output flow.

User Service Requirements:

USR = 1.0;
USR = 1.8.0;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(c);
USR = 1.8.1.3;
USR = 1.8.1.3(c);
USR = 1.8.1.4;
USR = 1.8.1.4(c);

Output Flow Dynamics Assumptions:

other_TRM_service_data = fotrm-transit_services;
totrm-transit_services = transit_services_for_other_TRM;

4.2.3.8 Provide Interface for Transit Service Raw Data

Input Flows

fmtsp_transit_service_data
map_transit_data
other_TRM_service_data
request_transit_service_external_data
request_transit_service_internal_data
request_transit_services_data_for_output
transit_routes_data
transit_schedule_data
transit_service_raw_data

Output Flows

transit_service_external_data
transit_service_internal_data
transit_service_raw_data
transit_services_data_for_output
transit_services_for_other_TRM

Description:

Overview: This process shall provide and manage the interface to the store in which the raw transit service data is held. This data shall be sent to the process by the routes and schedules generation processes, which are the only other processes permitted to access the store, and then in read-only mode. The received data shall be loaded into the store and distributed by this process to the three processes that are responsible for distributing the data within the transit center (TRM), to other local ITS functions, and to other transit centers (Other TRM), respectively. The process shall read data from the store and return it to whichever of the other three processes has made a data request. Data shall also be received by the process from other transit centers (Other TRM) and from multimodal transportation service providers. The process shall load this data into the data store for use by the local route and schedule generation processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fmtsp-transit_service_data';
(b) 'other_TRM_service_data';
(c) 'request_transit_services_data_for_output';
(d) 'request_transit_service_external_data';
(e) 'request_transit_service_internal_data';
(f) 'transit_routes_data';
(g) 'transit_schedule_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'map_transit_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_services_data_for_output';
(b) 'transit_service_external_data';
(c) 'transit_services_for_other_TRM';
(d) 'transit_service_internal_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either of the last two unsolicited input flows is received, automatically output each of the solicited output flows shown above;
(c) when either of the first two unsolicited data flows is received, the data shall be loaded into the data store transit services raw data;
(d) when any of the unsolicited inputs that request data is received, the requested data shall be read from the store using the solicited input flow listed above, and shall then be output to the requesting process using the relevant solicited output flow listed above;
(e) where required, the digitized data showing transit maps shall be included in the output flow generated in (d);
(f) manage the data in the store of raw transit service data.

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

transit_service_raw_data = ftfm-initiate_service_updates;
transit_service_external_data = ftfm-initiate_service_updates + request_transit_service_external_data;
transit_service_internal_data = ftfm-initiate_service_updates + request_transit_service_internal_data;
transit_services_for_other_TRM = ftfm-initiate_service_updates;
transit_services_data_for_output = request_transit_services_data_for_output;

4.2.3.9 Update Transit Map Data

Input Flows

fmup_transit_map_update
request_transit_map_update

Output Flows

map_data_for_transit
tmup_transit_map_update_request

Description:

Overview: This process shall provide updates to the store of digitized map data used by the transit route generation process and as the background for displays of transit services requested by the transit fleet manager. The process shall obtain the new data from a specialist data supplier or some other appropriate data source, after receiving an update request from the transit fleet manager interface process within the function. The processes requiring data for use in transit route generation and as the background to displays will read the data from the store loaded by this process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_transit_map_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to external functions:
(a) 'fmup-transit_map_update'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmup-transit_map_update_request';
(b) 'map_data_for_transit'.

Functional Requirements: This process shall meet the following requirements:
(a) continuously monitor for the receipt of the unsolicited data flow shown above;
(b) when the data flow in (a) is received, generate the first solicited output data flow shown above and continuously monitor for receipt of the solicited input data flow shown above;
(c) when the flow in (b) is received, output the second solicited output data flow shown above;
(d) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of digitized map data;
(e) manage the data in the store of digitized map data.

User Service Requirements:

USR = 1.8;
USR = 1.8.2;
USR = 1.8.2.4;
USR = 1.8.2.4(c);
USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.2.2;
USR = 2.1.2.2.2;

Output Flow Dynamics Assumptions:

tmup-transit_map_update_request = request_transit_map_update;
map_data_for_transit = fmup-transit_map_update;

4.2.4 Manage Transit Archive Data

Input Flows

bad_transit_collected_fare_payment
bad_transit_roadside_fare_payment
bad_transit_vehicle_fare_payment
ftso_archive_commands
paratransit_service_data_for_archive
transit_archive_request
transit_archive_status
transit_data_archive
transit_driver_info_for_archive
transit_emergency_data_for_archive
transit_fare_transactions
transit_incident_info_for_archive
transit_operational_data_for_archive
transit_route_assign_for_archive
transit_services_for_deployment
transit_technician_info
transit_user_payments_transactions
transit_vehicle_data_for_archive
transit_vehicle_maintenance_info

Output Flows

transit_archive_data
transit_data_archive
ttso_archive_status

Description:

Overview: This process shall obtain transit passenger and deployment data, transit user payment transaction data, transit emergency data, transit security data, maintenance and personnel data, and distribute it to the Manage Archive Data function. The process shall run when a request for data is received from an external source, or when fresh data is received.

All inputs to this process are unsolicited, and all outputs are solicited, except that the 'transit_archive_status' is a solicited input.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited data inputs shown above is received, the process shall store them in the data store along with meta data (data attributes about the data), and update the catalog;
(c) when the unsolicited input from the transit system operator is received, the process shall update the data store accordingly;
(d) when the request for transit archive data is received, the process shall immediately generate the solicited output shown above from the data store and send the data to the Manage Archived Data
function;
(e) the process should then receive the transit archive status solicited input and send this status to the transit system operator;
(f) if the status received in (e) was bad, the process shall attempt to correct the data and re-send it to the Manage Archived Data function;
(g) data shall only be sent to the source from which the data request originated;
(h) before output, the process shall put the data into a format that is easily read and interpreted by external processes and can also be read by travelers and transit users with the minimum of further processing.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.4;
USR = 7.1.3.1.4(a);
USR = 7.1.3.1.4(b);
USR = 7.1.3.1.4(d);
USR = 7.1.3.1.4(e);
USR = 7.1.3.1.4(f);

USR = 7.1.3.1.4(g);
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(b);
USR = 7.1.3.1.9(c);
USR = 7.1.3.1.9(d);
USR = 7.1.3.1.9(e);

Output Flow Dynamics Assumptions:

transit_archive_data = transit_archive_data_request;
ttso-archive_status = transit_archive_status;

4.3.1 Monitor Transit Vehicle Condition

Input Flows

transit_vehicle_maintenance_specs
transit_vehicle_status

Output Flows

transit_vehicle_maintenance
transit_vehicle_maintenance_information

Description:

Overview: This process shall monitor the condition of a transit vehicle. It shall use the transit vehicle maintenance specification to analyze brake, drive train, sensors, fuel, steering, tire, processor, communications equipment, and transit vehicle mileage to identify mileage based maintenance,
out-of-specification or imminent failure conditions. The data resulting from this analysis shall be loaded by the process into the store of transit vehicle operations data, through the output flow transit vehicle maintenance. This data is then sent to the process that generates transit vehicle maintenance schedules.

Data Flows: The input data flows are unsolicited and the output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow transit vehicle status;
(b) when the input in (a) is received, generate the outputs identified above using data obtained from the data store through the input flow transit vehicle maintenance specs.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;

Output Flow Dynamics Assumptions:

transit_vehicle_maintenance = 1/(60*60*24)*ITS_TRANSIT_VEHS+1/(60*60*24)*ITS_PTRANSIT_VEHS;
transit_vehicle_maintenance_information =

4.3.2 Generate Transit Vehicle Maintenance Schedules

Input Flows

transit_vehicle_maintenance_information

Output Flows

transit_vehicle_availability
transit_vehicle_maintenance_schedule
transit_vehicle_maintenance_schedule_data

Description:

Overview: This process shall generate transit vehicle maintenance schedules and includes what and when maintenance or repair is to be performed. Transit vehicle availability listings (current and forecast) shall also be generated by the process to support transit vehicle assignment planning. The maintenance and/or repair that is to be performed on the transit vehicle shall be scheduled by the process for a specific month, week, day(s), and hour(s). The availability of the transit vehicle that is also output by the process shall be based upon the transit vehicle maintenance schedule. The process shall load each transit vehicle maintenance schedule that it produces into the store of transit vehicle operations data, through the process that maintains this data store.

of the following:
(a) 'transit_vehicle_maintenance_schedule', which contains data subsequently written to the transit vehicle operations data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, produce the maintenance schedule including details of the work that is to be done and when it shall be done;
(c) when (b) is completed, generate the outputs identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;
USR = 2.1.3.1.3;
USR = 2.1.3.1.4;

Output Flow Dynamics Assumptions:

transit_vehicle_availability = 1/(60*60*24)*ITS_TRANSIT_VEHS+1/(60*60*24)*ITS_PTRANSIT_VEHS;
transit_vehicle_maintenance_schedule = 1/(60*60*24)*ITS_TRANSIT_VEHS
+1/(60*60*24)*ITS_PTRANSIT_VEHS;
transit_vehicle_maintenance_schedule_data = 1/(60*60*24)*ITS_TRANSIT_VEHS
+1/(60*60*24)*ITS_PTRANSIT_VEHS;

4.3.3 Generate Technician Work Assignments

Input Flows

ftfm_technician_information_request
ftfm_technician_information_updates
transit_technician_data
transit_vehicle_maintenance_schedule_data
transit_vehicle_maintenance_verification_results

Output Flows

transit_technician_data
transit_technician_info
transit_technician_work_assignment
ttfm_technician_information
ttmp_work_schedule

Description:

Overview: This process shall assign transit maintenance personnel to a transit vehicle maintenance schedule. The maintenance schedule shall be received from another process and shall define what and when maintenance repair is to be performed to a specific transit vehicle. The process shall base the personnel assignment upon details about the personnel obtained from the transit fleet manager and held

in a local data store. These details shall comprise personnel eligibility, work assignments, preferences and seniority. The process shall also provide these details to the transit fleet manager on request. When a work assignment has been generated, the process shall send it to the transit maintenance personnel and also to the process that monitors and verifies maintenance work activity. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit vehicle maintenance input is received, generate the maintenance schedule, using data in the store of transit maintenance technician data;
(c) on completion of (b), output the work assignment to the maintenance personnel and the process that monitors and verifies maintenance work activity using the data flows identified above;
(d) when a request for personnel data is received from the transit fleet manager, retrieve the requested data from the store and output it to the manager using the data flow identified above;
(e) manage the data in the store of transit technician data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;
USR = 2.1.3.1.3;
USR = 2.1.3.1.4;

Output Flow Dynamics Assumptions:

transit_technician_data = ftfm-technician_information_updates+ftfm-technician_information_request +transit_vehicle_maintenance_verification_results;
transit_technician_work_assignment = 1/(60*60*24)*TRANSIT_TECHS*TRANSIT_FLEETS;
ttfm-technician_information = 1/(60*60)*TRANSIT_FLEETS;

4.3.4 Monitor And Verify Maintenance Activity

Input Flows

transit_technician_work_assignment
transit_vehicle_maintenance_specs
transit_vehicle_status

Output Flows

transit_vehicle_maintenance_log_data
transit_vehicle_maintenance_verification_results

Description:

Overview: This process shall verify that the transit vehicle maintenance activities were performed correctly and that a time stamped maintenance log for record keeping was generated. The correctness of the maintenance activities shall be judged by the process against the transit vehicle's status, the maintenance personnel's work assignment, and the transit maintenance schedules produced by other processes. The process shall save a time stamped record of all the maintenance activities performed on the vehicle into the transit vehicle maintenance log.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input of vehicle status is received, compare it with the maintenance schedule read from the store of transit vehicle operations data, input through the transit vehicle maintenance specs data flow, and the maintenance personnel work assignments generated by another process;
(c) if the result is satisfactory, generate the outputs identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
ttmp-work_schedule = 1/(60*60*24)*TRANSIT_TECHS*TRANSIT_FLEETS; USR = 2.1.2;
transit_technician_info = transit_technician_data; USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;
USR = 2.1.3.1.5;

Output Flow Dynamics Assumptions:

transit_vehicle_maintenance_log_data = 1/(60*60*24)*TRANSIT_VEHS;
transit_vehicle_maintenance_verification_results = 1/(60*60*24)*TRANSIT_VEHS;

4.3.5 Report Transit Vehicle Information

Input Flows

ftfm_transit_vehicle_maintenance_information_request
ftfm_transit_vehicle_maintenance_specs
transit_vehicle_maintenance_data

Output Flows

transit_vehicle_maintenance_data_request
transit_vehicle_maintenance_specs_update
ttfm_transit_vehicle_maintenance_information

Description:

Overview: This process shall provide the transit fleet managers with the capability of requesting and receiving transit vehicle maintenance information. The process shall obtain the data for each request from the store of transit vehicle operations data, through the process that manages the data store, and shall produce the output to the transit fleet manager in an easily understood form. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ftfm-transit_vehicle_maintenance_specs';
(b) 'ftfm-transit_vehicle_maintenance_information_request'.

Solicited Input Processing: This process shall receive the following solicited input data flows:
(a) 'transit_vehicle_maintenance_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ttfm-transit_vehicle_maintenance_information';
(b) 'transit_vehicle_maintenance_data_request', which requests data from the transit vehicle operations data store, through the process that manages the data store;
(c) 'transit_vehicle_maintenance_specs_update'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the inputs are received, read the requested data from the store of transit vehicle operations data, through the process that manages the data store;
(c) when (b) is complete, generate the output to the transit fleet manager identified above;
(d) when the new transit vehicle maintenance specification data is received from the fleet manager, generate the output to the transit vehicle operations data store management process; this process subsequently sends the transit vehicle maintenance specs to another process.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;
USR = 2.1.3.1.5;

Output Flow Dynamics Assumptions:

transit_vehicle_maintenance_data_request = ftfm-transit_vehicle_maintenance_information_request;
transit_vehicle_maintenance_specs_update = ftfm-transit_vehicle_maintenance_specs;
ttfm-transit_vehicle_maintenance_information = ftfm-transit_vehicle_maintenance_information_request;

4.3.6 Update Transit Vehicle Information

Input Flows

ftmp_transit_vehicle_maintenance_updates

Output Flows

transit_vehicle_maintenance_data_update

Description:

Overview: This process shall provide the transit maintenance personnel with the capability to update transit vehicle maintenance information. The process shall send the data received from the transit maintenance personnel to the transit vehicle operations data store management process for use by other processes.

Data Flows: The input data flow is unsolicited. The output flow is solicited and contains data that is written to the store of transit vehicle operations data, by the process that manages the data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received from the transit maintenance personnel, load the received data into the store of transit vehicle operations data using the output flow identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;
USR = 2.1.3.1.5;

Output Flow Dynamics Assumptions:

transit_vehicle_maintenance_data_update = ftmp-transit_vehicle_maintenance_updates;

4.3.7 Manage Transit Vehicle Operations Data Store

Input Flows

transit_vehicle_maintenance
transit_vehicle_maintenance_data_request
transit_vehicle_maintenance_data_update
transit_vehicle_maintenance_log_data
transit_vehicle_maintenance_schedule
transit_vehicle_maintenance_specs_update
transit_vehicle_operations_data

Output Flows

transit_vehicle_maintenance_data
transit_vehicle_maintenance_info
transit_vehicle_maintenance_specs
transit_vehicle_operations_data

Description:

Overview: This process shall manage the store of transit vehicle operations data. It shall be able to load data it receives about vehicle maintenance into the store and provide that data on request to other processes.

Data Flows: The input data flow is unsolicited and all output flows are solicited with the exception of the following:
(a) 'transit_vehicle_operations data', which contains data written to and read from the transit vehicle operations data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs containing new data are received, load the data into the store;
(c) if the data in (b) contains new maintenance specifications, send them to the vehicle condition and verify matinenance activities processes;
(d) when the input containing requests for data is received, retrieve the required data from the store and send it to the requesting process;
(e) manage the data in the store of transit vehicle operations data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.3;
USR = 2.1.3.1;
USR = 2.1.3.1.2;
USR = 2.1.3.1.3;
USR = 2.1.3.1.4;
USR = 2.1.3.1.5;

Output Flow Dynamics Assumptions:

transit_vehicle_maintenance_data = transit_vehicle_maintenance_data_request;
transit_vehicle_maintenance_specs = transit_vehicle_maintenance_specs_update;
transit_vehicle_operations_data = transit_vehicle_maintenance_data_update+
transit_vehicle_maintenance+
transit_vehicle_maintenance_schedule+ transit_vehicle_maintenance_log_data+
transit_vehicle_maintenance_data_request+ transit_vehicle_maintenance_specs_update;
transit_vehicle_maintenance_info = 1/(60*60);

4.4.1.1 Manage Transit Security

Input Flows

emergency_request_transit_details
secure_area_surveillance_information
transit_media_incident_interface_parameters
transit_operator_security_action

Output Flows

emergency_acknowledge_transit_details
secure_area_broadcast_message
secure_area_monitoring_control
transit_incident_details
transit_incident_info_for_archive
transit_incident_information
transit_media_incident_information
transit_operator_incident_information

Description:

Overview: This process shall manage the security in the transit system by monitoring for potential incidents. Data shall be obtained by the process from a variety of sources and assessed for any security problems. Problems shall be passed by the process to the transit system operator for review and the required action. Information about incidents shall also be sent by this process to another process for output to the media, using interface parameters set up by the transit system operator.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following solicited input which is received as a result of output to another process:
(a) 'transit_operator_security_action'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) generate the corresponding output data flows;
(c) continuously monitor the traveler emergency details from the Remote Traveler Subsystem to determine
if
any incidents are taking place, and if so generate the appropriate outputs.

User Service Requirements:

USR = 2.0;
USR = 2.4.0;
USR = 2.4.1;
USR = 2.4.1.1;
USR = 2.4.1.2;
USR = 2.4.2;
USR = 2.4.2.1;
USR = 2.4.2.2;
USR = 2.4.4;
USR = 2.4.4.1;
USR = 2.4.4.2;

Output Flow Dynamics Assumptions:

transit_incident_details = emergency_request_transit_details;
transit_incident_information = emergency_request_transit_details;
transit_media_incident_information = emergency_request_transit_details;
transit_operator_incident_information = emergency_request_transit_details;
secure_area_broadcast_message = emergency_request_transit_details;
emergency_acknowledge_transit_details = emergency_request_transit_details;

4.4.1.2 Manage Transit Emergencies

Input Flows

ftu_emergency_request
transit_driver_emergency_request
transit_operator_request_acknowledge
transit_vehicle_location

Output Flows

transit_driver_emergency_acknowledge
transit_emergency_details
transit_emergency_information
transit_operator_emergency_request

Description:

Overview: This process shall support the management of emergencies that occur in the transit system by processing information received from transit vehicles. The process shall accept inputs from either the transit vehicle driver or a transit user, the latter through such interfaces as panic buttons, alarm switches, etc. The reported emergencies shall be sent to another process for action by the transit system operator and subsequently for output to the media. The process shall also send acknowledgment data to the process providing the interface to the transit driver.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following solicited input which is received as a result of output to another process:
(a) 'transit_operator_request_acknowledge'.
secure_area_monitoring_control = emergency_request_transit_details+ftso-video_camera_action_request;
Functional Requirements: This process shall meet the following functional requirements:
transit_incident_info_for_archive = 1/(60*60); (a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) after sending notification to the transit system operator process, if no response is received, then the output message to that process shall be repeated periodically until a response is received;
(d) the data about the emergency sent to the transit system operator shall include the transit vehicle location which is derived from separate input for the emergency data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.4;
USR = 2.1.4.1;
USR = 2.1.4.4;
USR = 2.4.0;
USR = 2.4.1;
USR = 2.4.1.2;
USR = 2.4.1.3;

Output Flow Dynamics Assumptions:

transit_driver_emergency_acknowledge = ftu-emergency_request+transit_driver_emergency_request;
transit_emergency_information = ftu-emergency_request+transit_driver_emergency_request;
transit_emergency_details = ftu-emergency_request+transit_driver_emergency_request;
transit_operator_emergency_request = ftu-emergency_request+transit_driver_emergency_request;

4.4.1.3 Provide Transit System Operator Security Interface

Input Flows

ftso_emergency_request_acknowledge
ftso_media_parameter_request
ftso_media_parameter_updates
ftso_security_action
ftso_video_camera_action_request
transit_media_interface_parameters
transit_operator_emergency_request
transit_operator_incident_information

Output Flows

transit_media_interface_parameters
transit_operator_request_acknowledge
transit_operator_security_action
ttso_emergency_request
ttso_media_parameters
ttso_potential_incidents_alarm
ttso_potential_security_problem
ttso_video_image_data

Description:

Overview: This process shall provide an interface for the transit system operator to identify and act upon potential information security problems and emergencies. This information shall be provided by other processes through input data flows. This process shall also provide the capability for the transit system operator to update parameters that control the output of data about the potential security problems to the media. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following solicited inputs:
(a) 'ftso-emergency_request_acknowledge', which is a result of output to the transit system operator;
(b) 'ftso-security_action', which is a result of output to the transit system operator;
(c) 'transit_media_interface_parameters', which is data written to or requested from the store of transit media interface parameters.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs providing information about emergencies are received, generate the output to the transit system operator identified above;
(c) as a result of (b), monitor for input of a response from the transit system operator using the input defined above;
(d) on receipt of the data in (c), generate the output to the process that provided the information about the emergency;
(e) when the data flow requesting the media interface parameters is received, read the data from the store and output it to the transit system operator;
(f) when the data flow with updates to the media interface parameters is received, load the data into the store of these parameters;
(g) manage the data in the store of transit media interface parameters.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.1;
USR = 2.1.1.2;
USR = 2.1.1.2.1;
USR = 2.1.2;
USR = 2.1.2.1;
USR = 2.1.2.1.2;
USR = 2.1.4;
USR = 2.1.4.1;
USR = 2.1.4.4;
USR = 2.4.0;
USR = 2.4.4;
USR = 2.4.4.1;

USR = 2.4.4.2;
USR = 2.4.4.3;

Output Flow Dynamics Assumptions:

transit_media_interface_parameters = ftso-media_parameter_updates+ftso-media_parameter_request;
transit_operator_security_action = 1/(60*60)*TRANSIT_FLEETS;
transit_operator_request_acknowledge = 1/(60*60)*TRANSIT_FLEETS;
ttso-emergency_request = 1/(60*60)*TRANSIT_FLEETS;
ttso-media_parameters = ftso-media_parameter_request;
ttso-potential_incidents_alarm = 1/(60*60)*TRANSIT_FLEETS;
ttso-potential_security_problem = 1/(60*60)*TRANSIT_FLEETS;
ttso-video_image_data = 1/(60*60)*TRANSIT_FLEETS;

4.4.1.4 Provide Transit External Interface for Emergencies

Input Flows

fm_transit_incident_information_request
transit_media_emergency_information
transit_media_incident_information

Output Flows

tm_transit_emergency_information
tm_transit_incident_information
transit_incident_data

Description:

Overview: This process shall provide the interface through which information about security problems and emergencies detected within the transit system are distributed directly to the media and other information systems. This process shall construct its output from the data supplied by other processes. This data shall contain parameters that define the way (format, content, etc.) in which the information is output by the process. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All inputs are unsolicited and all outputs are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) upon receipt of either of the inputs the process shall immediately generate the appropriate output, using the supplied parameters to determine the information and format in which it shall be supplied.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.4;
USR = 2.1.4.3;
USR = 2.1.4.4;
USR = 2.1.4.4(d);

Output Flow Dynamics Assumptions:

tm-transit_emergency_information = transit_media_incident_information;
tm-transit_incident_information = transit_media_emergency_information;
transit_incident_data = transit_media_incident_information;

4.4.1.5 Provide Transit Driver Interface for Emergencies

Input Flows

ftd_emergency_request
transit_driver_emergency_acknowledge

Output Flows

transit_driver_emergency_request
ttd_emergency_information

Description:

Overview: This process shall provide an interface to the transit vehicle through which the driver can both report an emergency situation and receive an acknowledgment. The process shall provide this interface in such a way that its operation for both inputs and outputs shall be transparent to transit users on board the vehicle and to anyone outside the vehicle, and shall not compromise the safe operation of the vehicle by the driver.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ftd-emergency_request'.

Solicited Input Processing: This process shall receive the following solicited input data flows:
(a) 'transit_driver_emergency_acknowledge'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_driver_emergency_request';
(b) 'ttd-emergency_information'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) upon receipt of the unsolicited input, immediately generate both outputs shown above;
(c) the solicited input should then be received;
(d) the output to the transit driver and the method of providing the input must be transparent to transit users and anyone in the vicinity of the transit vehicle.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.4;
USR = 2.1.4.2;
USR = 2.1.4.4;
USR = 2.1.4.4(a);
USR = 2.1.4.4(b);
USR = 2.1.4.4(c);
USR = 2.1.4.4(d);

Output Flow Dynamics Assumptions:

transit_driver_emergency_request = ftd-emergency_request;
ttd-emergency_information = transit_driver_emergency_acknowledge;

4.4.1.6 Collect Transit Vehicle Emergency Information

Input Flows

transit_emergency_details
transit_media_emergency_interface_parameters

Output Flows

transit_emergency_data
transit_emergency_data_for_archive
transit_media_emergency_information

Description:

Overview: This process shall collect data about emergencies that occur on-board transit vehicles for output to the media and the Manage Emergency Services function. These emergencies may be reported by
either the transit driver or a transit user, the latter through such interfaces as panic buttons, alarm switches, etc. For output to the media interface process, the data shall be combined with the data in the media interface parameters data store.

Data Flows: All input data flows are unsolicited and all output flows are solicited, with the exception of the following which contains data read from a data store:
(a) 'transit_media_emergency_interface_parameters'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input is received, generate the two outputs identified above, adding the data from the media interface parameters store to that being sent to the media interface process.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.4;
USR = 2.1.4.3;
USR = 2.1.4.4;
USR = 2.4.0;
USR = 2.4.1;
USR = 2.4.1.1;
USR = 2.4.1.1(a);
USR = 2.4.1.1(b);
USR = 2.4.1.1(c);
USR = 2.4.1.1(d);
USR = 2.4.1.1(e);
USR = 2.4.1.3;

Output Flow Dynamics Assumptions:

transit_emergency_data = transit_emergency_details;
transit_media_emergency_information = transit_emergency_details;
transit_emergency_data_for_archive = 1/(60*60);

4.4.1.7 Monitor Secure Area

Input Flows

fsa_area_image
secure_area_broadcast_message
secure_area_monitoring_control

Output Flows

secure_area_surveillance_information

Description:

Overview: This process shall monitor the secure area environment. Data shall be obtained by the process from a variety of sources and assessed for any security problems. Problems shall be passed by the process to other processes for review and the required action. Information about incidents shall also be sent by this process to another process for output to the media, using interface parameters set up by the transit system operator. The process shall also provide facilities for the control of video cameras and audio output in the secure area environment.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above that come from the secure area environment, or that are reporting traveler emergencies;
(b) when traveler/transit emergency details are received, generate the data flow that sends the incident information to the processes that pass it on to the transit system operator interface process;
(c) if no response is received from the output of the flow in (b), the output of the data flow in (b) shall be repeated periodically until a response is received;
(d) when a response is received to the data flow in (b), generate the other output data flows;
(e) continuously monitor the video (surveillance information) input flow from the secure area environment

to determine if any incidents are taking place, and if so carry out similar actions to those in (b) through (d) above;
(f) if the response data flow in (d) is received without there being any input from the secure area environment, output the monitoring control (camera, audio) data flow.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.4;
USR = 2.1.4.3;
USR = 2.1.4.4;
USR = 2.2.1.2;
USR = 2.2.1.2.1;
USR = 2.2.1.2.1.2;
USR = 2.2.1.2.1.2(a);
USR = 2.2.1.2.1.2(b);
USR = 2.2.1.2.1.2(c);
USR = 2.4.0;
USR = 2.4.1;
USR = 2.4.1.1;
USR = 2.4.1.1(a);
USR = 2.4.1.1(b);
USR = 2.4.1.1(c);
USR = 2.4.1.1(d);
USR = 2.4.1.1(e);
USR = 2.4.1.2;
USR = 2.4.2;
USR = 2.4.2.1;
USR = 2.4.2.2;
USR = 2.4.4;
USR = 2.4.4.1;
USR = 2.4.4.2;

Output Flow Dynamics Assumptions:

secure_area_surveillance_information = fsa-area_image+ft-traveler_secure_area_image;

4.4.1.8 Report Traveler Emergencies

Input Flows

emergency_acknowledge_transit_details
emergency_request_traveler_acknowledge
ft_remote_emergency_request

Output Flows

emergency_request_transit_details
emergency_request_traveler_details
tt_emergency_response

Description:

Overview: This process shall provide an interface in the Provide Driver and Traveler Services function through which travelers can declare emergencies. The traveler may be at a kiosk or other device, transit stop, transit depot, etc. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows from the traveler or Emergency Management listed above;
(b) when any of the inputs in (a) are received, check for content;
(c) generate the output identified above and send the data to the next process;
(d) when the emergency request input is received from the traveler, generation of the output to the Manage
Emergency Services function must take priority over all other processing;
(e) following the output of the message in (d), all other processing shall be suspended until the acknowledgment data flow is received from the Manage Emergency Services function, and the output has been
displayed.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.7.0;
USR = 1.7.1;
USR = 1.7.1.2;
USR = 1.7.1.2.1;
USR = 2.0;
USR = 2.1.0;
USR = 2.1.4;
USR = 2.1.4.3;
USR = 2.1.4.4;
USR = 2.4.1;
USR = 2.4.1.2;

Output Flow Dynamics Assumptions:

emergency_request_traveler_details = ft-remote_emergency_request;
emergency_request_transit_details =
ft-remote_emergency_request+secure_area_surveillance_information;

4.4.2 Coordinate Multiple Agency Responses to Incidents

Input Flows

ftfm_coordination_data
transit_emergency_information
transit_incident_coordination_data
transit_incident_information
transit_preplanned_responses_for_incidents

Output Flows

transit_coordination_data
ttfm_coordination_request

Description:

Overview: This process shall provide transit fleet managers with an interface through which they can control the coordination data sent to the Manage Emergency Services function following the detection of a security problem or emergency within the transit operations network by other processes. The process shall use data from the store of predefined responses to security problems and emergencies in the outputs that it sends to the Manage Emergency Services function. If no match can be found then the process shall send all the available data to the transit fleet manager for action. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'transit_emergency_information';
(b) 'transit_incident_information'.

Solicited Input Processing: This process shall receive the following solicited input data flows:
(a) 'transit_preplanned_responses_for_incidents', which contains data requested from a data store;
(b) 'ftfm-coordination_data', which is received as a result of a previous output to the transit fleet manager;
(c) 'transit_incident_coordination_data', which is received as a result of output being sent to processes in the Manage Emergency Services function.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_coordination_data';
(b) 'ttfm_coordination_request'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when either of the information inputs is received, search the store of predefined responses for a match;
(c) if (b) is successful, generate the output to the Manage Emergency Services function identified above;
(d) if (b) is unsuccessful, output all the data received as input to the transit fleet manager and monitor for receipt of the solicited input flow from the manager;
(e) when the solicited input flow in (d) is received, generate the output to the Manage Emergency Services function identified above;
(f) use the appropriate database mechanism(s) to retrieve data from the store of predefined responses identified above.

User Service Requirements:

USR = 2.0;
USR = 2.4.0;
USR = 2.4.4;
USR = 2.4.4.3;

Output Flow Dynamics Assumptions:

transit_coordination_data = 1/(60*60)*TRANSIT_FLEETS;
ttfm-coordination_request = 1/(60*60)*TRANSIT_FLEETS;

4.4.3 Generate Responses for Incidents

Input Flows

ftfm_request_response_parameter_output
ftfm_response_parameters
transit_preplanned_responses_for_incidents

Output Flows

transit_preplanned_responses_for_incidents
ttfm_response_parameter_output

Description:

Overview: This process shall provide the interface through which the transit fleet manager can enter and review predefined responses to security problems and emergencies that have been detected by other processes within the Manage Transit function. This data shall be stored in a form which can be used by another process to provide coordination data to the Manage Emergency Services function. The input and output forms shall include those that are suitable for travelers with physical disabilities.
inout flow contains data requested from or written to a data store:
(a) 'transit_preplanned_responses_for_incidents'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input requesting output of the store contents is received, read the data from the store of transit predefined responses;
(c) when (b) has been successfully completed, generate the output to the transit fleet manager identified above;
(d) when the input in (a) contains new transit predefined responses data, update the store, overwriting any old data as necessary;
(e) manage the data in the store of preplanned responses.

User Service Requirements:

USR = 2.4.4.5; USR = 2.0;
USR = 2.4.0;
USR = 2.4.4;
USR = 2.4.4.4;

Output Flow Dynamics Assumptions:

transit_preplanned_responses_for_incidents = 4/(60*60*24*7*52)*TRANSIT_FLEETS;
ttfm-response_parameter_output = 1/(60*60);

4.5.1 Assess Transit Driver Performance

Input Flows

transit_driver_performance_considerations

Output Flows

transit_driver_performance
transit_driver_performance_data

Description:

Overview: This process shall assess the transit driver's performance at previous work assignments. The process shall carry out this activity by 1) utilizing standardized performance evaluation criteria set forth by governmental regulations and transit operating company policies, 2) assessing the transit driver's driving history, and 3) assessing comments from the transit driver's supervisor(s). It shall also use the details of any moving violations or accidents, supervisor comments, government regulations, and company policies. The data shall be sent to this process by the process that provides the interface to a local data store, each time that the store is updated with driver performance data.

Solicited Input Processing: This process shall receive the following solicited input data flow:
(a) 'transit_driver_performance_considerations', contains data requested from a data store.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(a) 'transit_driver_performance', contains data written to a data store;
(b) 'transit_driver_performance_data', contains data sent to another process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flow listed above;
(b) when the input is received, analyze the data it contains and generate the two output flows identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.4;

Output Flow Dynamics Assumptions:

transit_driver_performance = transit_driver_performance_considerations;
transit_driver_performance_data = transit_driver_performance_considerations;

4.5.2 Assess Transit Driver Availability

Input Flows

transit_driver_availability_considerations

Output Flows

transit_driver_availability
transit_driver_availability_data

Description:

Overview: This process shall assess the transit driver's availability based on previous work assignments plus health and vacation commitments. The process shall carry out this activity by 1) utilizing standardized transit driver work criteria set forth by governmental regulations and company policies, 2) monitoring the transit driver's health status and vacation status, and 3) monitoring the transit driver's accumulated work hours. The data shall be sent to this process by the process that provides the interface to a local data store, each time that the store is updated with driver availability data.

(a) 'transit_driver_availability_considerations', contains data requested from a data store.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(b) 'transit_driver_availability', contains data written to a data store;
(c) 'transit_driver_availability_data', contains data sent to another process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flow listed above;
(b) when the input is received, analyze the data it contains and generate the two output flows identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.1;
USR = 2.1.3.2.2;
USR = 2.1.3.2.3;
USR = 2.1.3.2.4;
USR = 2.3.4;
USR = 2.3.4.3;

Output Flow Dynamics Assumptions:

transit_driver_availability = transit_driver_availability_considerations;
transit_driver_availability_data = transit_driver_availability_considerations;

4.5.3 Access Transit Driver Cost Effectiveness

Input Flows

transit_driver_cost_effectiveness_considerations

Output Flows

transit_driver_cost_effectiveness
transit_driver_cost_effectiveness_data

Description:

Overview: This process shall assess the transit driver's cost effectiveness when carrying out previous work assignments. The process shall perform this activity by 1) utilizing standard transit driver cost criteria set forth by governmental regulations and company policies, and 2) monitoring the transit driver's hourly wage and accumulated work hours. The data shall be sent to this process by the process that provides the interface to a local data store, each time that the store is updated with driver cost effectiveness data.

Solicited Input Processing: This process shall receive the following solicited input data flow:
(a) 'transit_driver_cost_effectiveness_considerations', contains data requested from a data store.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(a) 'transit_driver_cost_effectiveness', contains data written to a data store;
(b) 'transit_driver_cost_effectiveness_data', contains data sent to another process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flow listed above;
(b) when the input is received, analyze the data it contains and generate the two output flows identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.2;
USR = 2.1.3.2.4;

Output Flow Dynamics Assumptions:

transit_driver_cost_effectiveness = transit_driver_cost_effectiveness_considerations;
transit_driver_cost_effectiveness_data = transit_driver_cost_effectiveness_considerations;

4.5.4 Assess Transit Driver Eligibility

Input Flows

transit_driver_availability_data
transit_driver_cost_effectiveness_data
transit_driver_eligibility_considerations
transit_driver_performance_data

Output Flows

transit_driver_eligibility
transit_driver_eligibility_data

Description:

Overview: This process shall assess the transit driver's eligibility for future work assignments. The process shall carry out this activity by 1) monitoring the transit driver's performance, availability and cost effectiveness, 2) utilizing standardized transit driver eligibility criteria set forth by governmental regulations and company policies, and 3) ensuring that the transit driver has the required experience, education and certifications. The data shall be sent to this process in one of two ways: 1) by the process that provides the interface to a local data store, each time that the store is updated with driver eligibility data, or 2) the data is produced as the result of analysis work carried out by other processes within the Manage Traffic function.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following solicited flows that interface to the data store:
(a) 'transit_driver_eligibility_considerations', which contains data requested from a data store;
(b) 'transit_driver_eligibility', which contains data written to a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when any of the input flows are received, generate the outputs identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 3.1.3.2;
USR = 2.1.3.2.2;
USR = 2.1.3.2.4;

Output Flow Dynamics Assumptions:

transit_driver_eligibility = transit_driver_availability_data+transit_driver_cost_effectiveness_data
+transit_driver_eligibility_considerations+transit_driver_performance_data;
transit_driver_eligibility_data = transit_driver_eligibility_considerations
+transit_driver_availability_data+transit_driver_cost_effectiveness_data
+transit_driver_performance_data;

4.5.5 Generate Transit Driver Route Assignments

Input Flows

paratransit_services_for_transit_drivers
transit_driver_eligibility_data
transit_driver_route_assignment_considerations
transit_driver_route_data
transit_services_for_transit_drivers
transit_vehicle_availability

Output Flows

transit_driver_route_data
transit_route_assign_for_archive
ttd_route_assignements

Description:

Overview: This process shall assign transit drivers to transit schedules. The transit driver's eligibility, route preferences, seniority, and transit vehicle availability shall be used by the process to determine the transit driver's route assignment. The output produced by the process shall be sent to the transit driver in the form of the next work assignment. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when either of the inputs containing eligibility or route assignment consideration data is received, generate the output to the transit driver identified above;
(c) when any of the other inputs is received, load the data into a local data store for use in future route assignment calculations;
(d) manage the data in the store of transit driver route data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.2;
USR = 2.1.3.2.3;
USR = 2.1.3.2.3(a);
USR = 2.1.3.2.3(c);
USR = 2.1.3.2.3(d);
USR = 2.3.4;
USR = 2.3.4.3;

Output Flow Dynamics Assumptions:

ttd-route_assignements = transit_driver_eligibility_data
+transit_driver_route_assignment_considerations;
transit_route_assign_for_archive = 1/60;

4.5.6 Update Transit Driver Information

Input Flows

ftd_information_updates

Output Flows

transit_driver_consideration_inputs

Description:

Overview: This process shall provide the interface through which the transit driver can input data to the store of transit driver information. The interface provided by this process shall enable the transit driver to update personal availability and route assignment information. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: The input data flow is unsolicited and the output flow is solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow from the transit driver listed above;
(b) when the input is received, generate the output data flow identified above.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.4;

Output Flow Dynamics Assumptions:

transit_driver_consideration_inputs = ftd-information_updates;

4.5.7 Report Transit Driver Information

Input Flows

ftfm_transit_driver_information_request
ftfm_transit_driver_information_updates
ftfm_transit_driver_route_preferences
transit_driver_information_output

Output Flows

transit_driver_consideration_updates
transit_driver_information_output_request
ttfm_transit_driver_information

Description:

Overview: This process shall provide the interface between the transit fleet manager and the store of driver information. The interface provided by the process shall enable the fleet manager to review and update transit driver information. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: The input data flows are unsolicited and the output flows are solicited with the exception of the following solicited input:
(a) 'transit_driver_information_output',which is the result of a data request sent to transit driver information interface process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the transit fleet manager listed above;
(b) when any of the inputs in (a) is received, output the appropriate data flow to the interface process for the store of transit driver information identified above;
(c) if the input flow requested driver information, continuously monitor for receipt of the data flow sent from the data store interface process containing the requested data;
(d) when the input data flow in (c) is received, generate the appropriate output flow to the transit fleet manager, containing the requested data.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.4;

Output Flow Dynamics Assumptions:

transit_driver_consideration_updates = ftfm-transit_driver_information_updates
+ftfm-transit_driver_route_preferences;
transit_driver_information_output_request = ftfm-transit_driver_information_request;
ttfm-transit_driver_information = ftfm-transit_driver_information_request;

4.5.8 Provide Transit Driver Information Store Interface

Input Flows

transit_driver_availability
transit_driver_consideration_inputs
transit_driver_consideration_updates
transit_driver_cost_effectiveness
transit_driver_eligibility
transit_driver_information
transit_driver_information_output_request
transit_driver_performance

Output Flows

transit_driver_availability_considerations
transit_driver_cost_effectiveness_considerations
transit_driver_eligibility_considerations
transit_driver_info_for_archive
transit_driver_information
transit_driver_information_output
transit_driver_performance_considerations
transit_driver_route_assignment_considerations

Description:

Overview: This process shall provide the read and write interface to the store of transit driver information. The interface enables the contents of the store to be updated with inputs received from the transit driver and transit fleet manager via other processes, as well as, inputs resulting from analysis of driver availability, cost effectiveness, eligibility, and performance carried out by other processes. The process shall also supply data to these processes when the store is updated with information from the transit driver and fleet manager. It shall also supply data to the process that generates driver route assignments when any of the analysis inputs is received.

Data Flows: The input data flows are unsolicited and the output flows are solicited with the exception of the following solicited inputs:
(a) 'transit_driver_information', which contains data written to a data store;
(b) 'transit_driver_availability', which contains data received as a result of output being sent to the availability analysis process;
(c) 'transit_driver_cost_effectiveness', which contains data received as a result of output being sent to the cost effectiveness analysis process;
(d) 'transit_driver_eligibility', which contains data received as a result of output being sent to the eligibility analysis process;
(e) 'transit_driver_performance', which contains data received as a result of output being sent to the performance analysis process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when either the input or update data flow is received, load the data into the store of transit driver information and send the data to the appropriate analysis process;
(c) when the input containing the results of the availability, cost effectiveness, eligibility and performance analysis is received, again load it into the store of transit driver information and send the data to the driver route assignment process;
(d) when the input containing the request for output of the current store data is received from the transit fleet manager interface process, read the data from the store and send it to the requesting process;
(e) manage the data in the store of transit driver information.

User Service Requirements:

USR = 2.0;
USR = 2.1.0;
USR = 2.1.3;
USR = 2.1.3.2;
USR = 2.1.3.2.1;
USR = 2.1.3.2.2;
USR = 2.1.3.2.3;
USR = 2.1.3.2.4;

Output Flow Dynamics Assumptions:

transit_driver_availability_considerations = transit_driver_consideration_inputs;
transit_driver_cost_effectiveness_considerations = transit_driver_consideration_updates;
transit_driver_eligibility_considerations = transit_driver_consideration_updates;
transit_driver_information = transit_driver_consideration_updates+transit_driver_consideration_inputs;6
transit_driver_information_output = transit_driver_information_output_request;

transit_driver_performance_considerations = transit_driver_consideration_updates;
transit_driver_route_assignment_considerations = transit_driver_consideration_inputs;
transit_driver_info_for_archive = 1/60;

4.6.1 Detect Transit User on Vehicle

Input Flows

ftu_transit_user_vehicle_image
request_transit_user_vehicle_image
transit_user_vehicle_tag_data

Output Flows

transit_user_vehicle_image
transit_user_vehicle_tag_identity

Description:

Overview: This process shall detect embarking transit users on-board a transit vehicle and read data from the payment instrument that they are carrying. The process shall provide an image of all transit users which shall be used for violation processing of those who do not have a payment instrument or whose transit fare transaction fails. It shall obtain an image of the required accuracy under all lighting conditions and over the range of speeds with which transit users will pass through the fare collection point on a transit vehicle.

Data flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user tag data input flow is received, generate the transit user tag identity output flow identified above;
(c) when the flow requesting an image of the transit user is received, if necessary convert the video data in the flow from the transit user into a digital form, and output the digitized image in the transit user vehicle image data flow;
(d) if the input flow in (c) is not received, discard the video image data in the flow from the transit user;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the credit identity being transmitted using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.1;
USR = 3.1.2;
USR = 3.1.2.7;
USR = 3.1.2.8;
USR = 3.1.4;
USR = 3.1.4.3;

Output Flow Dynamics Assumptions:

transit_user_vehicle_image = request_transit_user_vehicle_image;
transit_user_vehicle_tag_identity = transit_user_vehicle_tag_data;

4.6.2 Determine Transit User Needs on Vehicle

Input Flows

transit_advisory_vehicle_information
transit_services_for_vehicle_fares
transit_user_vehicle_information
transit_user_vehicle_tag_identity
transit_vehicle_location

Output Flows

transit_user_vehicle_ride
transit_user_vehicle_ride_data

Description:

Overview: This process shall determine the transit user's travel routing based on the transit vehicle's current location and the user's destination. The process shall support the transit user's routing, enabling it to include travel on the vehicle for all or part of its route and (possibly) transfer to another vehicle on another route. In order to achieve this capability, the process shall have access to the complete range of transit services (routes and schedules) that are available to the transit user. The transit vehicle's location shall be provided by other processes within the Manage Transit function. Details of all transactions with the transit user's payment details removed, shall be sent by this process to the interface process for loading into a data store.

Data flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user tag identity input flow is received, continuously monitor for receipt of the flow with the other transit user information;
(c) when both the flows in (b) have been received, use the vehicle location and transit services inputs to generate the output flows identified above;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR - 3.1.2.7;
USR = 3.1.4;
USR = 3.1.4.3;

Output Flow Dynamics Assumptions:

transit_user_vehicle_ride_data = 1*ITS_TRANSIT_VEHS;
transit_user_vehicle_ride = 1*ITS_TRANSIT_VEHS;

4.6.3 Determine Transit Fare on Vehicle

Input Flows

transit_fares_for_vehicle
transit_user_vehicle_ride

Output Flows

transit_user_vehicle_fare

Description:

Overview: This process shall calculate the transit user's fare based on the origin and destination provided by the user. The process shall calculate the fare using the transit routing, transit fare category, and transit user history components of the ride data, in addition to information provided by the interface process for the transit fares data store. The accumulated data shall be sent by this process to another process for the actual implementation of the fare payment transaction.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'transit_fares_for_vehicle'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user ride data input flow is received, generate the output data flow identified above, using the data in the store of transit fares for vehicles;
(c) manage the data in the store of transit vehicle fares data;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.2;

Output Flow Dynamics Assumptions:

transit_user_vehicle_fare = 1*ITS_TRANSIT_VEHS;

4.6.4 Manage Transit Fare Billing on Vehicle

Input Flows

bad_tag_list_update
bad_transit_tag_store_data
confirm_vehicle_fare_payment
ftd_fare_transaction_mode_set_up
ftd_request_batch_mode_data_transfer
ftu_transit_user_vehicle_image
transit_user_transaction_queue
transit_user_vehicle_fare
transit_user_vehicle_tag_identity

Output Flows

bad_tag_list_request
bad_transit_tag_store_check_request
bad_transit_tag_store_updates
fare_collection_vehicle_violation_information
request_vehicle_fare_payment
transit_user_transaction_record
transit_user_vehicle_payment_response
transit_user_vehicle_processed_fare_data
ttd_batch_mode_data_transfer_status
ttd_request_fare_transaction_mode_set_up
ttu_vehicle_access_message

Description:

Overview: This process shall manage the transit user fare payments on-board a transit vehicle. The process shall receive information about the fare that is to be paid and the method of payment adopted by the transit user. It shall always support two modes of operation to complete the back end financial processing: infrastructure interactive, or semi-autonomous batch processing. The interactive method shall be used for individual transactions, such as those in paratransit type operations where value/volume ratios are high. It shall send transit user fare payment data to processes in the Provide Electronic Payment Services function for financial authorization and transaction processing, plus the return of the result for display to the transit user. A failed transaction shall result in the transmission of an image of the transit user to another process. Batch processing shall be used by the process for routes where value/volume ratios are low. It shall be performed using all the same data flows and processes as in the interactive method, except that transaction records are queued in a transaction buffer store which shall be maintained by this process. The accumulated data for the fare transactions shall be sent to the Provide Electronic Payment Services function on command from the transit vehicle driver, or when the transit vehicle has reached a convenient point on its route. The transit vehicle driver shall be notified when batch processing has completed successfully. In either mode of operation, a record of the status of all transit fare processing shall be sent to an interface process for the fare collection storage database.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) if the fare processing mode is not set, then request it as input from the transit vehicle driver;
(c) when confirmation of the mode is received, and at the end of each service, request an update to the store of bad tag data, completely overwriting the existing data with the new data;
(d) when the transit user fare input data is received, check the payment information against the store of bad tag data;
(e) if a match is found in (d), output a transaction failed message to the transit user;
(f) if no match is found in (d), generate the necessary outputs identified above that are consistent with the mode of processing being employed;
(g) when confirmation of an interactive mode transaction is received, pass on the result to the transit user;
(h) transmit the fare transaction data collected in batch mode when an instruction is received from the transit vehicle driver, the vehicle reaches the end of its current service, or the store of fare data becomes full;
(i) when confirmation of successful completion of batch mode fare processing is received, clear the fare data from the store and inform the transit vehicle driver;
(j) if the batch mode fare transaction fails, then inform the transit vehicle driver;
(k) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog

encryption techniques;
(l) manage the data in the stores of bad tag data and the transit user transaction buffer.

User Service Requirements:
USR = 2.3.0;

Output Flow Dynamics Assumptions:
bad_tag_list_request = transit_user_vehicle_fare;
bad_transit_tag_store_updates = 1/(60*60*24)*ITS_TRANSIT_VEHS;
bad_transit_tag_store_check_request = transit_user_vehicle_fare;
fare_collection_vehicle_violation_information = (transit_user_vehicle_fare)/10;
transit_user_vehicle_payment_response = transit_user_vehicle_fare;
request_vehicle_fare_payment = transit_user_vehicle_fare;
transit_user_vehicle_processed_fare_data = transit_user_vehicle_fare;
transit_user_transaction_record = transit_user_vehicle_fare*(PARATRANSIT_USERS/TRANSIT_USERS);
ttd-batch_mode_data_transfer_status = ftd-request_batch_mode_data_transfer;
ttd-request_fare_transaction_mode_set_up = 1/(60*60*24)*ITS_TRANSIT_VEHS;
ttu-vehicle_access_message = transit_user_vehicle_fare;

4.6.5 Provide Transit User Fare Payment Interface on Vehicle

Input Flows

transit_user_vehicle_credit_identity
transit_user_vehicle_payment_response
transit_vehicle_advanced_payment_response
transit_vehicle_location

Output Flows

transit_user_advanced_payment_on_vehicle
transit_user_vehicle_information
transit_vehicle_advanced_payment_request
ttu_vehicle_payment_confirmed

Description:

Overview: This process shall provide the fare payment interface for the transit user on-board a transit vehicle. The process shall prompt the transit user for information necessary that has not been provided for the transaction. The result of the transit service ride fare payment plus other services request and payment, shall be reported back to the transit user by the process. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs from the transit user are received, generate the appropriate outputs identified above, prompting the user for any information that has not been supplied;
(c) when any response flow is received, generate the appropriate output to the transit user to indicate the success or failure of the requested transaction;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 2.2.1.2.2;
USR = 2.2.1.2.2.1;
USR = 2.2.1.2.2.2;
USR = 2.2.1.2.2.3;
USR = 2.2.1.2.2.4;
USR = 2.3.0;
USR = 2.3.3;
USR = 2.3.3.1;
USR = 2.3.3.1(c);
USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.3;

Output Flow Dynamics Assumptions:

transit_user_advanced_payment_on_vehicle = 1*ITS_TRANSIT_VEHS;
transit_user_vehicle_information = 1*ITS_TRANSIT_VEHS;
transit_vehicle_advanced_payment_request = 10/60*ITS_TRANSIT_VEHS;
ttu-vehicle_payment_confirmed = 1*ITS_TRANSIT_VEHS;

4.6.6 Update Transit Vehicle Fare Data

Input Flows

transit_vehicle_fare_data

Output Flows

transit_fares_for_vehicle

Description:

Overview: This process shall provide a database on-board the transit vehicle for use in fare processing. The database shall contain transit fare information from which the fares for all possible trips within the transit operational network can be determined.

Data Flows: The input data flow is unsolicited and the output flow contains data written to a data store:
(a) 'transit_fares_for_vehicle'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input flow is received, generate the output flow identified above to update the contents of the store of transit fares;
(c) manage the data in the store of transit fares.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.6;

Output Flow Dynamics Assumptions:

transit_fares_for_vehicle = transit_vehicle_fare_data;

4.6.7 Provide Transit Vehicle Passenger Data

Input Flows

transit_user_vehicle_processed_fare_data
transit_user_vehicle_ride_data
transit_vehicle_fare_collection_data

Output Flows

transit_vehicle_fare_collection_data
transit_vehicle_passenger_data

Description:

Overview: This process shall provide passenger loading and fare statistics data to other ITS functions. The process shall send the data automatically at regular periodic intervals using data collected in the store of fare transaction data. This store receives data from the process that interfaces to the user on-board a transit vehicle.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from and written to a data store:
(a) 'transit_vehicle_fare_collection_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, write the data into the store of collected transit fare data;
(c) at periodic intervals, read all the data from the store and send it to the other processes in the Manage Transit function using the passenger data flow identified above;
(d) manage the data in the store of collected transit fare data;
(e) the data contained in the store shown above shall not contain any reference to a transit user's identity or credit payment information.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.2;
USR = 3.1.2.7;

Output Flow Dynamics Assumptions:

transit_vehicle_passenger_data = TRANSIT_STOPS/TRANSIT_WAIT_TIME;

4.6.8 Manage Transit Vehicle Advanced Payments

Input Flows

advanced_tolls_and_charges_vehicle_confirm
transit_vehicle_advanced_payment_request

Output Flows

advanced_tolls_and_charges_vehicle_request
transit_vehicle_advanced_payment_response

Description:

Overview: This process shall act as the interface for advanced payment of tolls and parking lot charges from the transit user. Requests for these advanced payments shall be passed to other processes in the Provide Electronic Payment Services function for transaction processing. The process shall ensure that the response to these requests from transit users is returned to the transit vehicle from which it was made.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the request input data flow is received from the transit user interface process, generate the request output data flow identified above;
(c) as a result of (b), continuously monitor for receipt of the confirmation flow identified above;
(d) when the flow in (c) is received, output the response flow identified above to the transit user interface process;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.3;

Output Flow Dynamics Assumptions:

transit_vehicle_advanced_payment_response = transit_vehicle_advanced_payment_request;
advanced_tolls_and_charges_vehicle_request = transit_vehicle_advanced_payment_request;

4.7.1.1 Provide Transit User Roadside Data Interface

Input Flows

ftu_transit_information_request
transit_services_for_travelers
transit_services_roadside_data
transit_vehicle_arrival_time

Output Flows

transit_services_roadside_data
transit_services_travelers_request
ttu_transit_information

Description:

Overview: This process shall communicate with the Transit Management Center (TRMC) by providing public transit information at roadside locations. These locations may consist of transit vehicle stops or other locations that provide general public transit information. The process shall enable the roadside unit to obtain information about the transit services on request from the local transit user interface process and to receive data about late running services from other processes within the Manage Transit function. The received data shall be loaded into a local data store for future use. The input and output forms shall include those that are suitable for travelers with physical disabilities.

(a) 'transit_vehicle_arrival_time';
(c) 'ftu-transit_information_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'transit_services_for_travelers';
(b) 'transit_services_roadside_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_services_roadside_data';
(b) 'transit_services_travelers_request';
(c) 'ttu-transit_information'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) periodically send the output flow requesting transit services for travelers and check for the response input flow;
(c) Send all data received to the local store 'transit_services_roadside_data';
(d) when the input request from the transit user is received, read the data in the local data store to see if the information is already present;
(e) if the required information is not present, generate the output flow requesting transit services for travelers and check for the response input flow;
(f) when the data flow in (e) is received, or if the data was read from the local data store, generate the output flow to the transit user with the requested data;
(g) manage the data in the store of transit services roadside data.

User Service Requirements:

USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.2;
USR = 2.2.1.2.1;
USR = 2.2.1.2.1.1;
USR = 2.2.1.2.1.1.1;
USR = 2.2.1.2.1.1.2;
USR = 2.2.1.2.1.1.2(a);
USR = 2.2.1.2.1.1.2(b);
USR = 2.2.1.2.1.1.3;
USR = 2.2.1.2.1.2;
USR = 2.2.1.2.1.3;
USR = 2.2.1.2.2;
USR = 2.2.1.2.2.1;
USR = 2.2.1.2.2.2;
USR = 2.2.1.2.2.3;
USR = 2.2.1.2.2.4;

Output Flow Dynamics Assumptions:

transit_information_transit_user_request = ftu-transit_information_request;
ttu-transit_information = ftu-transit_information_request + transit_vehicle_arrival_time;

4.7.1.2 Provide Transit User Roadside Vehicle Data Interface

Input Flows

transit_services_roadside_data
transit_vehicle_user_data

Output Flows

transit_services_roadside_data
ttu_transit_vehicle_information

Description:

Overview: This process shall provide the roadside (transit stop) interface through which transit users receive information about an approaching transit vehicle or one that has already arrived. The process shall output the data to the transit user as soon as it is received and shall load the data into the local store for future use. Output of the data shall be maintained until the vehicle leaves the stop, when the process shall cease output of the data and delete it from the local store. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'transit_vehicle_user_data'.

Solicited Input Processing: This process shall receive the following data flow as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'transit_services_roadside_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'transit_services_roadside_data';
(b) 'ttu-transit_vehicle_information'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input data flow is first received, calculate the time before the transit vehicle will arrive using the difference between the transit vehicle arrival time and the current time;
(c) output the data calculated in (b) to the transit user and store the data locally to provide a continuous source of data;
(d) once the data shows that the transit vehicle is leaving the stop, clear the output for that vehicle and delete its data from the local data store;
(e) if the input ceases to be received from the transit vehicle, maintain the output using the locally stored data;
(f) manage the data in the store of transit services roadside data.

User Service Requirements:

USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.2;
USR = 2.2.1.2.1;
USR = 2.2.1.2.1.1;
USR = 2.2.1.2.1.1.1;
USR = 2.2.1.2.1.1.2;
USR = 2.2.1.2.1.1.3;
USR = 2.2.1.2.1.2;
USR = 2.2.1.2.1.3;
USR = 2.2.1.2.2;
USR = 2.2.1.2.2.1;
USR = 2.2.1.2.2.2;
USR = 2.2.1.2.2.3;

Output Flow Dynamics Assumptions:

transit_information_for_transit_users = transit_information_transit_user_request;
transit_services_roadside_data = transit_services_for_travelers+transit_vehicle_arrival_time;
transit_services_travelers_request = transit_information_transit_user_request;
ttu-transit_vehicle_information = transit_services_roadside_data + transit_vehicle_user_data;

4.7.2.1 Detect Transit User at Roadside

Input Flows

ftu_transit_user_roadside_image
request_transit_user_roadside_image
transit_user_roadside_tag_data

Output Flows

transit_user_roadside_image
transit_user_roadside_tag_identity

Description:

Overview: This process shall detect transit users embarking at a roadside transit stop and read data from the payment instrument that they are carrying. The process shall provide an image of all transit users which shall be used for violation processing of those who do not have a payment instrument or whose transit fare transaction fails. It shall obtain an image of the required accuracy under all lighting conditions and over the range of speeds with which transit users will pass through the fare collection point at the roadside, i.e., a transit stop.

Data flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user tag data input flow is received, generate the transit user tag identity output flow identified above;
USR = 2.2.1.2.2.4; (c) when the flow requesting an image of the transit use is received, if necessary convert the video data in the flow from the transit user into a digital form, and output the digitized image in the transit user vehicle image data flow;
(d) if the input flow in (c) is not received, discard the video image data in the flow from the transit user;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the credit identity being transmitted using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.1;
USR = 3.1.2;
USR = 3.1.2.7;
USR = 3.1.2.8;
USR = 3.1.4;
USR = 3.1.4.3;

Output Flow Dynamics Assumptions:

transit_user_roadside_image = request_transit_user_roadside_image;
transit_user_roadside_tag_identity = transit_user_roadside_tag_data;

4.7.2.2 Determine Transit User Needs at Roadside

Input Flows

transit_services_for_roadside_fares
transit_user_roadside_information
transit_user_roadside_tag_identity

Output Flows

transit_user_roadside_ride
transit_user_roadside_ride_data

Description:

Overview: This process shall determine the transit user's travel routing based on the user's destination and the location of the roadside transit stop from which the route request is being made. The process shall support the transit user's routing enabling it to include travel on all or part of the route(s) operating from the stop and (possibly) transfer to another route. In order for this to be achieved, the process requires access to the complete range of transit services (routes and schedules) that are available to the transit user. Details of all transactions with the transit user's payment details removed, shall be sent by this process to the interface process for loading into the transit roadside fare collection data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user tag identity input flow is received, continuously monitor for receipt of the flow with the other transit user information;
(c) when both the flows in (b) have been received, use the vehicle location and transit services inputs to generate the output flows identified above;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR - 3.1.2.7;
USR = 3.1.4;
USR = 3.1.4.3;

Output Flow Dynamics Assumptions:

transit_user_roadside_ride_data = 1*ITS_TRANSIT_VEHS;
transit_user_roadside_ride = 1*ITS_TRANSIT_VEHS;

4.7.2.3 Determine Transit Fare at Roadside

Input Flows

transit_fares_for_roadside
transit_user_roadside_ride

Output Flows

transit_user_roadside_fare

Description:

Overview: This process shall calculate the transit user's fare based on the origin and destination provided by the user. The process shall calculate the fare using the transit routing, transit fare category, and transit user history components of the ride data together with data provided by the interface process to the database of transit fares. The accumulated data shall be sent by the process to another process for the actual implementation of the fare payment transaction.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'transit_fares_for_roadside'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user ride data input flow is received, generate the output data flow identified above, using the data in the store of transit fares for roadside;
(c) manage the data in the store of transit fares for roadside;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.2;

Output Flow Dynamics Assumptions:

transit_user_roadside_fare = 1*ITS_TRANSIT_VEHS;

4.7.2.4 Manage Transit Fare Billing at Roadside

Input Flows

confirm_roadside_fare_payment
ftu_transit_user_roadside_image
transit_user_roadside_fare
transit_user_roadside_tag_identity

Output Flows

fare_collection_roadside_violation_information
request_roadside_fare_payment
transit_user_roadside_payment_response
transit_user_roadside_processed_fare_data
ttu_roadside_access_message

Description:

Overview: This process shall generate the data necessary to enable the financial transaction between the transit user and the transit provider to be completed at the roadside, i.e., at a transit stop.
The process shall accept and process current transit passenger fare collection information. The process shall perform the front end transaction between the transit user and the transit system, and use the infrastructure interactive mode of operation to complete the back end processing. This means that the process shall send data about each transaction to processes in the Provide Electronic Payment Services function for the back end financial authorization and transaction processing. The process shall then await the return of the result for display to the transit user before accepting the next transaction. A failed transaction shall result in the transmission of an image of the transit user to another process. A record of the status of all transit fare processing shall be sent to another process for storage in a fare collection database.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the transit user fare input is received, generate the necessary outputs identified above that are consistent with the type of processing being employed;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.2;
USR = 3.1.2.3;
USR = 3.1.2.4;
USR = 3.1.2.5;
USR = 3.1.2.6;
USR = 3.1.2.7;

Output Flow Dynamics Assumptions:

fare_collection_roadside_violation_information = TRANSIT_USERS_PER_STOP/100;
transit_user_roadside_payment_response = transit_user_roadside_fare;
request_roadside_fare_payment = transit_user_roadside_fare;
transit_user_roadside_processed_fare_data = transit_user_roadside_fare;
ttu-roadside_access_message = transit_user_roadside_fare;

4.7.2.5 Provide Transit User Roadside Fare Interface

Input Flows

advanced_tolls_and_charges_roadside_confirm
ftu_destination_at_roadside
ftu_other_services_roadside_request
other_services_roadside_response
transit_user_roadside_credit_identity
transit_user_roadside_payment_response

Output Flows

advanced_tolls_and_charges_roadside_request
other_services_roadside_request
transit_user_advanced_payment_at_roadside
transit_user_roadside_information
ttu_other_services_roadside_confirmed
ttu_roadside_payment_confirmed

Description:

Overview: This process shall provide the interface for the transit user at the roadside, i.e., at a transit stop. The interface shall enable the transit user to specify the required destination of a transit service ride and request other (yellow pages) services. The process shall prompt the transit user for information necessary for the transaction that has not been provided. The result of the transit service ride fare payment plus other services request and payment, shall be reported back to the transit user by the process. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs from the transit user are received, generate the appropriate outputs identified above, prompting the user for any information that has not been supplied;
(c) when either of the response flows is received, generate the appropriate output to the transit user to indicate the success or failure of the requested transaction;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.3;

Output Flow Dynamics Assumptions:

advanced_tolls_and_charges_roadside_request = 10/60*ITS_TRANSIT_VEHS;
other_services_roadside_request = 12/(60*60)*ITS_TRANSIT_VEHS;
transit_user_advanced_payment_at_roadside = 1*ITS_TRANSIT_VEHS;
transit_user_roadside_information = 1*ITS_TRANSIT_VEHS;
ttu-other_services_roadside_confirmed = 12/(60*60)*ITS_TRANSIT_VEHS;
ttu-roadside_payment_confirmed = 1*ITS_TRANSIT_VEHS;

4.7.2.6 Update Roadside Transit Fare Data

Input Flows

transit_roadside_fare_data

Output Flows

transit_fares_for_roadside

Description:

Overview: This process shall provide a database at the roadside, i.e., a transit stop, for use in fare processing. The database shall contain transit fare information from which the fares for all possible trips within the transit operational network can be determined.

Data Flows: The input data flow is unsolicited and the output flow contains data written to a data store:
(a) 'transit_fares_for_roadside'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow listed above;
(b) when the input flow is received, generate the output flow identified above to update the contents of the data store of transit fares;
(c) manage the data in the store of transit fares for roadside.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.6;

Output Flow Dynamics Assumptions:

transit_fares_for_roadside = transit_roadside_fare_data;

4.7.2.7 Provide Transit Roadside Passenger Data

Input Flows

transit_roadside_fare_collection_data
transit_user_roadside_processed_fare_data
transit_user_roadside_ride_data

Output Flows

transit_roadside_fare_collection_data
transit_roadside_passenger_data

Description:

Overview: This process shall create passenger loading and fare statistics data based upon data collected at the roadside and send this data to the store of transit operations data. The process may send the data at regular periodic intervals, on-demand, or through some other trigger mechanism. The process shall create its outputs using information collected in the store of fare transaction data. This data is received from other processes at the roadside, i.e., at a transit stop.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from and written to a data store:
(a) 'transit_fare_collection_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, write the data into the store of collected transit fare data;
(c) at periodic intervals, read all the data from the data store and send it to the other processes in the Manage Transit function using the passenger data flow identified above;
(d) manage the data in the store of collected transit fare data;
(e) the data contained in the store shown above shall not contain any reference to a transit user's identity or credit payment information.

User Service Requirements:

USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.2;
USR = 3.1.2.7;

Output Flow Dynamics Assumptions:

transit_roadside_passenger_data = 1/(60*60*24)*TRANSIT_STOPS;

5.1.1 Identify Emergencies from Inputs

Input Flows

cf_hazmat_route_information
emergency_request_personal_traveler_details
emergency_request_traveler_details
fep_planned_event_data
fets_caller_information
fets_incident_information
incident_alert
mayday_emergency_data
transit_emergency_data
transit_incident_details

Output Flows

tep_planned_event_confirmation
verified_emergency

Description:

Overview: This process shall enable existing emergency centers to receive the calls, determine response requirements (enough to determine what responding agencies to notify), and route distress calls to those predesignated responding agencies. This process shall provide the identified emergency information in a standard format as required. This process receives emergency requests from the general public, public safety agencies, and other service providers (e.g., a Mayday service provider). Every set of emergency data received shall be assigned a level of confidence by the process depending on its source, so that the subsequent processes can assess the level of response to be provided. This process shall include verification, in that it shall determine if a number of inputs might all be referring to the same incident, then designate that incident in its notifications to the most appropriate responding agencies. By reconciling numerous reports and other collaborative information from the field (e.g.,CCTV images, reports from field staff), the verification function confirms the existence, location, and nature of a reported emergency.

Data Flows: All inputs are unsolicited and all outputs are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the inputs are received, the process shall perform an analysis of the data to produce the output in a standard format;
(c) the data format produced in (b) shall include a classification of the level of confidence or probability that the data is accurate, i.e., that it relates to a 'real' emergency and the information is correct.

User Service Requirements:

USR = 4.5;
USR = 4.5.0;
USR = 4.5.3;
USR = 4.5.3.1;
USR = 5.0;
USR = 5.1;
USR = 5.2;

Output Flow Dynamics Assumptions:

verified_emergency = ERMS_CALLS;
tep-planned_event_confirmation = ERMS_CALLS;

5.1.2 Determine Coordinated Response Plan

Input Flows

emergency_service_allocation_data
foec_emergency_center_identity
foec_incident_details
foec_incident_response_coordination
verified_emergency

Output Flows

emergency_response_data_for_communications
emergency_response_data_for_management
emergency_service_allocation_data_request
toec_emergency_center_identity
toec_incident_details
toec_incident_response_coordination

Description:

Overview: This process shall determine the appropriate response for a verified emergency. This process shall classify, prioritize, and respond to verified emergencies accordingly. This process shall also determine the appropriate response plan and activate any remote controlled functions requested by a basic_vehicle terminator through the vehicle_security_status data flow. A detailed description of the emergency, and any request for remote controlled emergency system activity, and any suggested response plan shall be sent to other processes for implementation. The same information shall also be forwarded to other emergency centers (other EM) for information and possible action.

Data Flows: All inputs are unsolicited with the exception of emergency_service_allocation_data which is a solicited flow from the data store. All outputs are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, the response will be determined from the data requested from the interface process that manages the store of emergency service allocation criteria and any functions requested by a vehicle shall be activated;
(c) when (b) is complete, the data shall be sent to both the emergency management and communications processes.

User Service Requirements:

USR = 5.0;
USR = 5.1;
USR = 5.2;

Output Flow Dynamics Assumptions:

emergency_response_data_for_communications = ERMS_CALLS;
emergency_response_data_for_management = ERMS_CALLS;
emergency_service_allocation_data_request = ERMS_CALLS;
toec-incident_details = ERMS_CALLS;
toec-emergency_center_identity = ERMS_CALLS;
toec-incident_response_coordination = ERMS_CALLS;

5.1.3 Communicate Emergency Status

Input Flows

detailed_emergency_status
emergency_response_data_for_communications
emergency_service_action_log
emergency_service_log_output_request
fm_emergency_information_request
incident_details_request
incident_information_request

Output Flows

emergency_data_request
emergency_request_driver_acknowledge
emergency_request_personal_traveler_acknowledge
emergency_request_traveler_acknowledge
emergency_request_vehicle_acknowledge
emergency_service_log_for_archive
emergency_service_log_output
incident_details
incident_information
tets_incident_acknowledge
tm_emergency_information
transit_incident_coordination_data

Description:

Overview: This process shall receive the emergency service response plans and the status of their implementation for dissemination to other ITS functions. That dissemination shall be subject to sanitization according to pre-arranged rules, implemented in this process. The process shall also read data about emergency responses from the emergency services action log. All data shall be communicated by the process in standard formats to travelers, drivers, and other ITS functions. In the case of in-vehicle, personal traveler, and transit emergencies, after each emergency becomes a verified incident, the data shall be sent as soon as new status or plan data is received. Dissemination shall be controlled according to rules determined in this process to limit the information transmitted to that information useful to the receiver. Emergency information that is received from the emergency telephone system or E911 operators, shall be disseminated only when the response plan data is first received. That has the effect of only disseminating data on incidents that have been verified, since only verified incidents will have response plans. The process shall also extract data from the emergency service action log on request from processes in other ITS functions, and from the emergency services operator. Communication to in-vehicle processes may include requests for additional information or a set of commands to the vehicle security system.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following solicited input which contains data requested from a data store:
(a) 'emergency_service_action_log'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either the input of emergency response plan data or detailed emergency status is received, read any associated data from the emergency service action log data store and generate the outputs to the vehicle, traveler, and Manage Transit function identified above;
(c) if the input in (b) is the first notification for an emergency, and it was received from the emergency telephone or E911 operator, send the acknowledge message identified above;
(d) when any of the input flows requesting information is received, read the data from the emergency service action log, and send it in the data flow of incident information identified above to the requesting process;
(e) manage the data in the store of the emergency service action log.

User Service Requirements:

USR = 5.0;
USR = 5.1;
USR = 5.2;

Output Flow Dynamics Assumptions:

emergency_request_driver_acknowledge = emergency_request_driver_details;
emergency_request_traveler_acknowledge = emergency_request_traveler_details;
emergency_request_personal_traveler_acknowledge = emergency_request_personal_traveler_details;
emergency_request_vehicle_acknowledge = emergency_request_vehicle_details;
emergency_service_log_output = emergency_service_log_output_request;
incident_details = incident_details_request;
incident_information = incident_information_request;
tets-incident_acknowledge = ERMS_CALLS;
tm-emergency_information = detailed_emergency_status;
transit_incident_coordination_data = transit_incident_details;

```
emergency_data_request = ERMS_CALLS;
emergency_service_log_for_archive = 1/(60*60);
```

5.1.4 Manage Emergency Response

Input Flows

cf_hazmat_vehicle_information
emergency_response_data_for_management
emergency_service_allocation_override
emergency_vehicle_acknowledge
emergency_vehicle_dispatch_status
fws_current_weather
fws_predicted_weather
incident_command_request
incident_response_clear
incident_status_data
incident_video_for_emergency_services
resource_deployment_status
traffic_data_for_emergency_services
transit_coordination_data
wrong_way_vehicle_detection

Output Flows

cf_hazmat_request
detailed_emergency_status
emergency_service_action_log
emergency_service_allocations
emergency_vehicle_dispatch_failure
emergency_vehicle_incident_details
emergency_vehicle_response_request
incident_response_status
local_decision_support
remote_video_image_control
resource_request

Description:

Overview: This process shall enable existing emergency centers to receive emergency calls, determine response requirements to the extent necessary to route the information, and route distress calls and emergency information to predesignated responding agencies and vehicles. All identified emergency information shall be provided by the process in a standard format as required. The process shall also communicate with commercial fleet managers to obtain details of cargo and other vehicle data where this

will affect the response of the emergency services, e.g., in the case of a vehicle carrying a HAZMAT load.

The current status of all emergency service responses shall be stored by the process in an action log, for access by the communications process.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:

1. Output to a data store:
(a) 'emergency_service_action_log'.
2. Solicited Input Processing:
(a) 'cf_hazmat_vehicle_information', which is received as a result of output to a process in the Manage Commercial Vehicles function;
(b) 'emergency_vehicle_dispatch_status', which is received as a result of output to another process in the Emergency Management Subsystem;
(c) 'resource_deployment_status', which is received as a result of output to another process in the Emergency Management Subsystem.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input of emergency response plan data is received, generate the output data flows identified above, and create an initial entry in the emergency response action log data store;
(c) when other inputs are received, update the data for the emergency to which they relate in the emergency service action log data store;
(d) if the emergency vehicle dispatch status indicates a failure, send the data to the action log and to the emergency services operator interface process;
(e) manage the data in the store of the emergency service action log.

User Service Requirements:

USR = 4.5;
USR = 4.5.0;
USR = 4.5.2;
USR = 4.5.2.1;
USR = 4.5.2.2;
USR = 4.5.2.3;
USR = 4.5.2.3(a);
USR = 4.5.2.3(b);
USR = 4.5.2.3(c);
USR = 4.5.2.3(e);
USR = 4.5.2.3(f);
USR = 4.5.2.3(g);
USR = 4.5.2.3(h);
USR = 4.5.3;
USR = 4.5.3.3;
USR = 4.5.3.4;
USR = 5.0;
USR = 5.2;
USR = 5.2.1;
USR = 5.2.1.1;
USR = 5.2.1.2;
USR = 5.2.1.3;
USR = 5.2.2;
USR = 5.2.2.1;

Output Flow Dynamics Assumptions:

emergency_service_action_log = ERMS_CALLS;
emergency_service_allocations = ERMS_CALLS;
emergency_vehicle_dispatch_failure = ERMS_CALLS/20;
emergency_vehicle_response_request = ERMS_CALLS;
emergency_vehicle_incident_details = ERMS_CALLS;
detailed_emergency_status = ERMS_CALLS;
cf_hazmat_request = 5/(60*60*24);
incident_response_status = 12/(60*60);
local_decision_support = ERMS_CALLS;
wrong_lane_violation_dection = ERMS_CALLS;
resource_request = ERMS_CALLS;
remote_video_image_control = 12/(60*60);

5.1.5 Manage Emergency Service Allocation Store

Input Flows

emergency_service_allocation_criteria
emergency_service_allocation_data_output_request
emergency_service_allocation_data_request
emergency_service_allocation_data_updates

Output Flows

archive_provide_emergency_service_allocation_data
emergency_service_allocation_criteria
emergency_service_allocation_data
emergency_service_allocation_data_output

Description:

Overview: This process shall manage the store of data that defines the way in which the emergency service resources shall be deployed in response to emergencies. Deployment shall vary by certain criteria, such as, type of emergency, source of information, time of day, location, etc. Parameters to define this allocation shall be loaded into the data store following receipt from the process that provides the emergency services operator interface.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'emergency_service_allocation_data_output_request';
(b) 'emergency_service_allocation_data_request';
(c) 'emergency_service_allocation_data_updates'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval:
(a) 'emergency_service_allocation_criteria', which is data retrieved from a data store.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'emergency_service_allocation_criteria', which is data written to a data store;
(b) 'emergency_service_allocation_data';
(c) 'emergency_service_allocation_data_output'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the inputs are received, generate the appropriate outputs identified above to read to or write data from the data store of emergency service allocation criteria;
(c) manage the data in the store of emergency service allocation criteria.

User Service Requirements:

USR = 2.0;
USR = 2.2;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.4;
USR = 2.3;
USR = 2.3.4;

Output Flow Dynamics Assumptions:

emergency_service_allocation_data_output = emergency_service_allocation_data_output_request;
emergency_service_allocation_criteria = 1/(60*60*24*7);
emergency_service_allocation_data = ERMS_CALLS;
archive_provide_emergency_service_allocation_data = 1/(60*60*24);

5.1.6 Process Mayday Messages

Input Flows

emergency_request_driver_details
emergency_request_vehicle_details
foec_mayday_emergency_data
mayday_vehicle_tracking

Output Flows

mayday_emergency_data
mayday_vehicle_tracking
toec_mayday_emergency_data

Description:

Overview: This process shall receive mayday messages from vehicles and drivers, determine whether the mayday message indicates an emergency that requires the attention of public safety agencies, and forward mayday emergency data to the appropriate agency when assistance is required. The content of the data flow 'mayday emergency data' shall include all the key data from the incoming data flow 'emergency request details' and an agency ID indicating the mayday provider that received and processed the mayday message. While not depicted in the logical architecture, the process will also be heavily dependent on voice communications to better ascertain the nature and severity of the emergency and to report this information to the appropriate local agency. This process shall also receive and keep a historical log of signals sent in the tracking_vehicle data flow.

Data Flows: All inputs are unsolicited with the exception of mayday_vehicle_tracking which is solicited along with all outputs.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the inputs are received, the process shall perform an analysis of the data to produce the output in a standard format;
(c) the data format produced in (b) shall include a classification of the level of confidence or probability that the data is accurate, i.e., that it relates to a 'real' emergency and the information is correct.

User Service Requirements:

USR = 4.5;
USR = 4.5.0;
USR = 4.5.3;
USR = 4.5.3.1;
USR = 5.0;
USR = 5.1;
USR = 5.2;

Output Flow Dynamics Assumptions:

mayday_emergency_data = ERMS_CALLS;
toec-mayday_emergency_data = ERMS_CALLS;

5.2 Provide Operator Interface for Emergency Data

Input Flows

emergency_service_allocation_data_output
emergency_service_allocations
emergency_service_log_output
emergency_vehicle_dispatch_failure
feso_emergency_action_log_request
feso_emergency_allocation_override
feso_emergency_data_input
feso_emergency_data_output_request
feso_emergency_display_update_request
map_data_for_emergency_display

Output Flows

emergency_service_allocation_data_output_request
emergency_service_allocation_data_updates
emergency_service_allocation_override
emergency_service_log_output_request
request_emergency_display_update
teso_emergency_action_log_output
teso_emergency_data_output
teso_emergency_vehicle_dispatch_failure

Description:

Overview: This process shall provide the emergency services operator with an interface to the other processes in the Manage Emergency Services function. The process shall enable the operator to review and update the data used to allocate emergency services to incidents, applying temporary overrides to current emergency service allocations to suit the special needs of a current incident, and requesting output of the log of emergency service actions. It shall also enable the output of a message showing the failure of an emergency vehicle dispatched in response to an incident. This output shall override all other outputs. The process shall support inputs from the emergency services operator in both manual and audio form, and shall provide its outputs in audible and visual forms. The visual output may appear in either hardcopy or as a display, or both, and an audible output shall accompany the emergency vehicle dispatch failure message.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the emergency services operator and the emergency vehicle dispatch failure flow listed above;
(b) when the inputs from the emergency services operator are received, send the appropriate output data flows to other processes;
(c) when the responses to the flows generated in (b) are received, send the appropriate outputs to the emergency services operator;
(d) when the emergency vehicle dispatch failure flow is received, generate the appropriate output flow listed above, overriding all other output flows.

User Service Requirements:

USR = 5.0;
USR = 5.1;
USR = 5.2;

Output Flow Dynamics Assumptions:

emergency_service_allocation_data_output_request = feso-emergency_data_output_request;
emergency_service_allocation_override = feso-emergency_allocation_override;
emergency_service_allocation_data_updates = feso-emergency_data_input;
emergency_service_log_output_request = feso-emergency_data_output_request;
request_emergency_display_update = feso-emergency_display_update_request;
teso-emergency_data_output = feso-emergency_data_output_request;
teso-emergency_action_log_output = feso-emergency_data_output_request;
teso-emergency_vehicle_dispatch_failure = .1 * feso-emergency_data_output_request;

5.3.1 Select Response Mode

Input Flows

emergency_vehicle_incident_details
emergency_vehicle_response_request
emergency_vehicle_status_data_for_responses

Output Flows

emergency_vehicle_dispatch_data
emergency_vehicle_dispatch_status
emergency_vehicle_status_data_change
emergency_vehicle_status_data_request

Description:

Overview: This process shall select the appropriate emergency services and their vehicle(s) to respond to incidents. The process shall determine the type and number of vehicles to be dispatched, and provide the vehicle(s) with information on the type and location of the incident. It shall request data about the vehicles that are available from the interface process to the data store of emergency vehicle status. Once the vehicle determination has been made, the status data shall be changed by the process, and incident data sent to the process responsible for the actual dispatch of the vehicle(s).

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'emergency_vehicle_response_request';
(b) 'emergency_vehicle_incident_details'.

Solicited Input Processing: This process shall receive the following solicited input data flow as a result of requesting information from another process:
(a) 'emergency_vehicle_status_data_for_responses'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'emergency_vehicle_status_data_change', an update command for a store (managed by another process);
(b) 'emergency_vehicle_dispatch_status';
(c) 'emergency_vehicle_status_data_request';
(d) 'emergency_vehicle_dispatch_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the two vehicular inputs identified above;
(b) when the emergency vehicle response request input data flow is received, request data from the process providing the interface for the store of emergency vehicle status;
(c) use the data obtained in (b) to determine the vehicle(s) necessary to provide the appropriate response to the incident;
(d) when the vehicle(s) have been determined in (c), generate the output identified above to dispatch each vehicle;
(e) send changed emergency vehicle status data back to the data store interface process to reflect the new vehicle status resulting from output of the dispatch data;
(f) when all the required vehicle type(s) and numbers have been dispatched, send the emergency vehicle dispatch status data flow to the process responsible for managing emergency responses, showing that the dispatch was successful;
(g) if no vehicles of the required type(s) are available, send the emergency vehicle dispatch status data flow to the process responsible for managing emergency responses, showing that the dispatch has failed for the particular vehicle type(s).

User Service Requirements:

USR = 5.0;
USR = 5.2;
USR = 5.2.1;
USR = 5.2.1.2;
USR = 5.2.1.3;

Output Flow Dynamics Assumptions:

emergency_vehicle_dispatch_data = emergency_vehicle_dispatch_request;
emergency_vehicle_dispatch_status = emergency_vehicle_dispatch_request;
emergency_vehicle_status_data_request = emergency_vehicle_dispatch_request;
emergency_vehicle_status_data_change = emergency_vehicle_dispatch_request;

5.3.2 Dispatch Vehicle

Input Flows

emergency_traffic_control_response
emergency_vehicle_dispatch_data
emergency_vehicle_dispatch_response
emergency_vehicle_route
emergency_vehicle_status_data_for_dispatch

Output Flows

emergency_traffic_control_request
emergency_vehicle_dispatch_request
emergency_vehicle_route_request
emergency_vehicle_suggested_route

Description:

Overview: This process shall direct selected emergency vehicles and drivers to respond to an incident, receive acknowledgment that they will in fact respond, and provide them with the location and details of the incident that was pre-calculated and sent to this process.
If called for, the process shall send details to the Manage Traffic function to request a
traffic control preemption be provided for the vehicle(s) if that mode of priority is available and chosen.
The
data for the emergency vehicle driver shall be sent to the driver interface process.

Data Flows: All inputs are unsolicited with the exception of emergency_traffic_control_response and emergency_vehicle_route which are solicited as are all outputs.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the dispatch and status data input flows listed above;
(b) when the flows in (a) are received, generate the outputs identified above to request the emergency vehicle route and provide the driver with information about the incident, monitoring for the receipt of any reply data;
(c) when the emergency vehicle route data is received, generate the emergency traffic control request data and send it to the Manage Traffic function.

User Service Requirements:

USR = 1.6;
USR = 1.6.3;
USR = 1.6.3.2;
USR = 1.6.3.2.2;
USR = 1.6.3.2.2(c);
USR = 4.5;
USR = 4.5.0;
USR = 4.5.3;
USR = 4.5.3.2;
USR = 5.0;
USR = 5.2;
USR = 5.2.1;
USR = 5.2.1.2;
USR = 5.2.1.3;
USR = 5.2.2;

Output Flow Dynamics Assumptions:

emergency_vehicle_dispatch_request = ERMS_CALLS*ACT_ERMS_VEHS;
emergency_traffic_control_request = ERMS_CALLS*ACT_ERMS_VEHS;
emergency_vehicle_route_request = ERMS_CALLS*ACT_ERMS_VEHS;
emergency_vehicle_suggested_route = ERMS_CALLS*ACT_ERMS_VEHS;

5.3.3 Track Vehicle

Input Flows

vehicle_location_for_emergency_services

Output Flows

emergency_vehicle_preemptions
emergency_vehicle_tracking_data

Description:

Overview: This process shall manage information about the location of all emergency vehicles available for dispatch and that have been dispatched, and the ETA for vehicles en route. The process shall send this data to the store of emergency vehicle status data. If the vehicle is on its way to an emergency, as indicated by the received vehicle status, the process shall also send data to processes in the Manage Traffic function that will enable the vehicle to have whatever level and mode of priority is available and granted at traffic signals.

Data Flows: All input data flows are unsolicited and the output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when the location flow is received and if it is different from the previous value, generate the data flow to the emergency vehicle status store interface process, adding the current date and time to the received location data;
(c) when the status flow is received, if it shows that the vehicle is on its way to an emergency incident, then simultaneously with (b) output the data flow requesting local vehicle priority to the indicator control processes in the Manage Traffic function.

User Service Requirements:

USR = 5.0;
USR = 5.2;
USR = 5.2.1;
USR = 5.2.1.1;

Output Flow Dynamics Assumptions:

emergency_vehicle_preemptions = 1;
emergency_vehicle_tracking_data = 1*ERMS_VEHS;

5.3.4 Assess Response Status

Input Flows

emergency_vehicle_route_assignment
emergency_vehicle_status_data_for_assessment
incident_status_update

Output Flows

emergency_vehicle_acknowledge
emergency_vehicle_status_data_needed
emergency_vehicle_status_data_update

Description:

Overview: This process shall assess the status of emergency vehicles that are responding to an incident. In making its assessment, the process shall use data from the process managing a store of vehicle status, plus data from the emergency vehicle driver interface process. The process shall send the results of the assessment to the process responsible for managing emergency and emergency response

information and update the store of vehicle status.

Data Flows: All input data flows are unsolicited and the output flows are solicited with the exception of the following solicited input flow, which contains data requested from the process managing the data store of emergency vehicle status data:
(a) 'emergency_vehicle_status_data_for_assessment';

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input from the driver is received, use the data it contains to request the relevant emergency vehicle status data from the store interface process;
(c) update the status using the input received from the driver and generate the flow to the process that manages emergency responses;
(d) send the revised emergency vehicle status data to the store interface process.

User Service Requirements:

USR = 5.0;
USR = 5.2;
USR = 5.2.1;

Output Flow Dynamics Assumptions:

emergency_vehicle_status_data_needed = emergency_driver_status_update;
emergency_vehicle_status_data_update = emergency_driver_status_update;
emergency_vehicle_acknowledge = emergency_driver_status_update;

5.3.5 Provide Emergency Personnel Interface

Input Flows

emergency_vehicle_dispatch_request
emergency_vehicle_suggested_route
fep_emergency_dispatch_acknowledge
fep_incident_command_request
fep_incident_status
local_decision_support

Output Flows

emergency_vehicle_dispatch_response
incident_command_request
incident_status_data
incident_status_update
tep_decision_support
tep_emergency_dispatch_order

Description:

Overview: This process shall provide an interface for emergency personnel, through which data can be exchanged with other processes in the Manage Emergency Services function. It shall support the exchange of incident data to which responses are being made by emergency personnel. The process shall support inputs from emergency personnel in both audible and manual forms, with outputs being
available in both audio or visual forms. The visual form may include display and hardcopy formats. Both inputs and outputs shall be provided in such a way that while alerting the driver to the information they contain, they shall in no way impair the driver's ability to operate the vehicle in a safe manner.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received from emergency personnel, generate the appropriate output flows identified above and send them to the dispatch and response status monitoring processes;
(c) when input is received from the dispatch process, generate the corresponding output to emergency personnel.

User Service Requirements:

USR = 5.0;
USR = 5.2;
USR = 5.2.2;
USR = 5.2.2.2;

Output Flow Dynamics Assumptions:

incident_status_update = fep-emergency_dispatch_acknowledge+fep-incident_status;
emergency_vehicle_dispatch_response = fep-emergency_dispatch_acknowledge;
tep-emergency_dispatch_order =
emergency_vehicle_dispatch_request+emergency_vehicle_suggested_route;
incident_command_request = fep-incident_command_request;
tep-decision_support = local_decision_support;
incident_status_data = fep-emergency_dispatch_acknowledge+fep-incident_status;

5.3.6 Maintain Vehicle Status

Input Flows

emergency_vehicle_status_data
emergency_vehicle_status_data_change
emergency_vehicle_status_data_needed
emergency_vehicle_status_data_request
emergency_vehicle_status_data_update
emergency_vehicle_tracking_data

Output Flows

archive_manage_emergency_vehicle_data
emergency_vehicle_status_data
emergency_vehicle_status_data_for_assessment
emergency_vehicle_status_data_for_dispatch
emergency_vehicle_status_data_for_responses

Description:

Overview: This process shall maintain a data store of the current status of all emergency vehicles available for dispatch and that have been dispatched. It shall provide data from the store on request from other processes and shall update the contents of the store with new data received from other processes. The process shall output the status of a vehicle to the process responsible for vehicle tracking for as long as it is on its way to an incident, to update ETA estimates and enable local vehicle priority to be given at intersections, if that mode of priority is chosen and granted.

Data Flows: The input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for the receipt of the input flows listed above;
(b) when either the data request or data needed flows is received, read the requested data from the store and send it to the requesting process;
(c) when either of the data flows containing updated or changed vehicle status data is received, load the new data into the data store, overwriting the existing data for the vehicle;
(d) when the new status data shows that a vehicle is on its way to an incident, output the vehicle status data to the processes responsible for vehicle tracking and vehicle dispatch;
(e) when the new status data shows that a vehicle is no longer on its way to an incident, output the status data to the process responsible for vehicle tracking, but do not send any further status updates until the condition in (d) is again satisfied;
(f) manage the data in the store of emergency vehicle status data, retrieving or writing individual records for one or more emergency vehicles, as required.

User Service Requirements:

USR = 5.0;
USR = 5.2;
USR = 5.2.1;
USR = 5.2.1.1;

Output Flow Dynamics Assumptions:

emergency_vehicle_status_data = emergency_vehicle_status_data_change
+emergency_vehicle_status_data_update;
emergency_vehicle_status_data_for_dispatch = emergency_vehicle_status_data_change;
emergency_vehicle_status_data_for_assessment = emergency_vehicle_status_data_needed;
emergency_vehicle_status_data_for_responses = emergency_vehicle_status_data_request;
archive_manage_emergency_vehicle_data = 1/60;

5.3.7 Provide Emergency Vehicle Route

Input Flows

emergency_vehicle_route_request
fmup_emergency_route_map_update

Output Flows

emergency_vehicle_route
emergency_vehicle_route_assignment
tmup_emergency_route_map_request

Description:

Overview: This process shall calculate and assign emergency vehicle routes for incident assistance upon request.

Data Flows: The inputs are unsolicited and the outputs are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the dispatch and status data input flows listed above;
(b) when the flows in (a) are received, generate the outputs identified above to request the emergency vehicle route and provide the driver with information about the incident, monitoring for the receipt of any reply data;
(c) when the emergency vehicle route data is received, generate the emergency traffic control request data
and send it to the Manage Traffic function.

User Service Requirements:

USR = 1.6;
USR = 1.6.3;
USR = 1.6.3.2;
USR = 1.6.3.2.2;
USR = 1.6.3.2.2(c);
USR = 4.5;
USR = 4.5.0;
USR = 4.5.3;
USR = 4.5.3.2;
USR = 5.0;
USR = 5.2;
USR = 5.2.1;
USR = 5.2.1.2;
USR = 5.2.1.3;
USR = 5.2.2;

Output Flow Dynamics Assumptions:

emergency_vehicle_route = 1/(60*60);
emergency_vehicle_route_assignment = 1/(60*60);
tmup-emergency_route_map_request = 1/(60*60*24);

5.4.1 Process TM Detected Violations

Input Flows

enforcement_data_for_TM
fdmv_traffic_violation_state_identity
fdmv_traffic_violation_vehicle_registration
hov_lane_violation
vehicle_pollution_alert

Output Flows

enforcement_data_for_TM
tdmv_traffic_violation_identity_code
tdmv_traffic_violation_vehicle_license
tea_traffic_violation_data

Description:

Overview: This process shall manage the details of high occupancy vehicle (HOV) lane use, wrong-way vehicle detection in reversible lanes, and pollution violations reported by the Manage Traffic function. The process shall use the parameters in the store of traffic management (TM) violation (enforcement) data to obtain the vehicle registration data from the appropriate State Department of Motor Vehicles (DMV) office, before sending all of the received information to the correct law enforcement agency. This process shall also maintain the TM enforcement data store, entering all information received from other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'hov_lane_violation';
(b) 'vehicle_pollution_alert'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_TM'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the DMV terminator:
(a) 'fdmv-traffic_violation_state_identity';
(b) 'fdmv-traffic_violation_vehicle_registration'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tdmv-traffic_violation_identity_code';
(b) 'tdmv-traffic_violation_vehicle_license';
(c) 'tea-traffic_violation_data';
(d) 'enforcement_data_for_TM'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either unsolicited input is received, use this data to generate the output requests to the DMV terminator, and enter this data in the local data store;
(c) when the solicited input flows from the DMV terminator have been received, use the data received in (b) and the contents of the TM data store to generate the traffic violation output to the law enforcement agency, and enter the new data in the local data store;
(d) manage the data in the store of traffic management violation data.

User Service Requirements:

USR = 1.0;
USR = 1.6;
USR = 1.6.0;
USR = 1.6.2;
USR = 1.6.2.4;
USR = 1.6.2.4.1;
USR = 1.6.3;
USR = 1.6.3.2;
USR = 1.6.3.2.2;
USR = 1.6.3.2.2(b);

Output Flow Dynamics Assumptions:

tdmv-traffic_violation_identity_code = hov_lane_violation + vehicle_pollution_alert;
tdmv-traffic_violation_vehicle_license = hov_lane_violation + vehicle_pollution_alert;
tea-traffic_violation_data = hov_lane_violation + vehicle_pollution_alert;

5.4.2 Process Violations for Tolls

Input Flows

enforcement_data_for_tolls
fdmv_toll_violation_state_identity
fdmv_toll_violation_vehicle_registration
toll_violation_information

Output Flows

enforcement_data_for_tolls
tdmv_toll_violation_identity_code
tdmv_toll_violation_vehicle_license
tea_toll_violation_data

Description:

Overview: This process shall manage the details of toll payment violations reported by the Provide Electronic Payments Services function. The process shall use the parameters in the store of toll payment violation (enforcement) data to obtain the vehicle registration data from the appropriate State Department of Motor Vehicles (DMV) office (or alternate source) for vehicles that are not equipped with a tag, before sending all of the received information to the correct law enforcement agency.
This process shall also maintain the toll payment enforcement datastore, entering all information received

from other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'toll_violation_information'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_tolls'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the DMV terminator:
(a) 'fdmv-toll_violation_state_identity';
(b) 'fdmv-toll_violation_vehicle_registration'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tdmv-toll_violation_identity_code';
(b) 'tdmv-toll_violation_vehicle_license';
(c) 'tea-toll_violation_data';
(d) 'enforcement_data_for_tolls'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the unsolicited input is received, use this data to generate the output requests to the DMV terminator;
(c) when the solicited input flows from the DMV terminator have been received, use the other data received in (b) and the contents of the toll payment violation (enforcement) data store to generate the toll violation output to the law enforcement agency, and enter the new data in the local data store;
(d) manage the data in the store of toll payment violation data.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;

Output Flow Dynamics Assumptions:

tdmv-toll_violation_identity_code = toll_violation_information;
tdmv-toll_violation_vehicle_license = toll_violation_information;
tea-toll_violation_data = toll_violation_information;

5.4.3 Process Parking Lot Violations

Input Flows

enforcement_data_for_parking
fdmv_parking_lot_violation_state_identity
fdmv_parking_lot_violation_vehicle_registration
parking_lot_violation_information

Output Flows

enforcement_data_for_parking
tdmv_parking_lot_violation_identity_code
tdmv_parking_lot_violation_vehicle_license
tea_parking_violation_data

Description:

Overview: This process shall manage the details of parking lot payment violations reported by the Provide Electronic Payment Services function. The process shall use the parameters in the store of parking lot violation (enforcement) data to obtain the vehicle registration data from the appropriate State Department of Motor Vehicles (DMV) office (or alternate source) for vehicles that are not equipped with a tag, before sending all of the received information to the correct law enforcement agency. This process shall also maintain the store of parking lot violation (enforcement) data, entering all information received from other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'parking_lot_violation_information'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_parking'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the DMV terminator:
(a) 'fdmv-parking_lot_violation_state_identity';
(b) 'fdmv-parking_lot_violation_vehicle_registration'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tdmv-parking_lot_violation_identity_code';
(b) 'tdmv-parking_lot_violation_vehicle_license';
(c) 'tea-parking_violation_data';
(d) 'enforcement_data_for_parking'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the unsolicited input shown above is received, use this data to generate the output requests to the DMV terminator;
(c) when the solicited input flows from the DMV terminator have been received, use the other data received in (b) and the contents of the parking lot payment violation data store to generate the parking lot violation output to the law enforcement agency, and enter the new data in the local data store;
(d) manage the data in the store of parking lot payment violation data.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;

Output Flow Dynamics Assumptions:

tdmv-parking_lot_violation_identity_code = parking_lot_violation_information;
tdmv-parking_lot_violation_vehicle_license = parking_lot_violation_information;
tea-parking_violation_data = parking_lot_violation_information;

5.4.4 Process Fare Payment Violations

Input Flows

enforcement_data_for_fare_payment
fare_violation_information

Output Flows

bad_transit_collected_fare_payment
enforcement_data_for_fare_payment
tea_fare_payment_violation_data

Description:

Overview: This process shall manage the details of fare payment violations reported by the Provide Electronic Payments function. The process shall use the parameters in the store of fare payment violation (enforcement) data to process and send the data to the correct law enforcement agency. This process shall also maintain the fare payment enforcement data store, entering all information received from other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'fare_violation_information'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_fare_payment'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above input being received:
(a) 'tea-fare_payment_violation_data';
(b) 'enforcement_data_for_fare_payment'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
USR = 3.1.1.4; (b) when the unsolicited input is received, use this data and the contents of the data store to generate the fare payment violation data to be sent to the law enforcement agency, and enter the new data in the local data store;
(c) manage the data in the store of enforcement data for fare payment violations.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;

Output Flow Dynamics Assumptions:

tea-fare_payment_violation_data = fare_violation_information;
bad_transit_collected_fare_payment = 1;

5.4.5 Process Vehicle Fare Collection Violations

Input Flows

enforcement_data_for_vehicle_fare_collection
fare_collection_vehicle_violation_information

Output Flows

bad_transit_vehicle_fare_payment
enforcement_data_for_vehicle_fare_collection
tea_fare_collection_vehicle_violation_data

Description:

Overview: This process shall manage the details of fare collection violations reported by the Manage Transit function that have taken place on-board a transit vehicle. The process shall use the parameters in the store of vehicle fare collection violation (enforcement) data to process and send the information to the correct law enforcement agency. This process shall also maintain the vehicle fare collection enforcement data store, entering all information received from other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'fare_collection_vehicle_violation_information'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_vehicle_fare_collection'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tea-fare_collection_vehicle_violation_data';
(b) 'enforcement_data_for_vehicle_fare_collection'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the unsolicited input is received, use this data and the contents of the data store to generate the vehicle fare collection violation data to be sent to the law enforcement agency, and enter the new data in the local data store;
(c) manage the data in the store of enforcement data for vehicle fare collection.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;

Output Flow Dynamics Assumptions:

tea-fare_collection_vehicle_violation_data = fare_collection_vehicle_violation_information;
bad_transit_vehicle_fare_payment = 1;

5.4.6 Process CV Violations

Input Flows

cv_violation_data
enforcement_data_for_cv
fdmv_cv_violation_state_identity
fdmv_cv_violation_vehicle_registration

Output Flows

enforcement_data_for_cv
tdmv_cv_violation_identity_code
tdmv_cv_violation_vehicle_license
tea_cv_violation_data

Description:

Overview: This process shall manage the details of violations committed by commercial vehicles, their drivers and/or operators, reported by the Manage Commercial Vehicles function. The process shall use the parameters in the store of commercial vehicle violation (enforcement) data to obtain the vehicle registration data from the appropriate State Department of Motor Vehicles (DMV) office, before sending all of the received data to the correct law enforcement agency. This process shall also maintain the commercial vehicle violation (enforcement) data store.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'cv_violation_data'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_cv'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the DMV terminator:
(a) 'fdmv-cv_violation_state_identity';
(b) 'fdmv-cv_violation_vehicle_registration'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'enforcement_data_for_cv';
(b) 'tdmv-cv_violation_identity_code';
(c) 'tdmv-cv_violation_vehicle_license';
(d) 'tea-cv_violation_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the unsolicited input is received, use the data it contains to generate the output requests to the DMV terminator;
(c) when the solicited input flows from the DMV terminator have been received, use the other data received in (b) and the contents of the store of commercial vehicle (cv) violation (enforcement) data to generate the the cv violation output to the law enforcement agency;
(d) manage the data in the store of commercial vehicle violation data.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;

Output Flow Dynamics Assumptions:

tdmv-cv_violation_identity_code = cv_violation_data;
tdmv-cv_violation_vehicle_license = cv_violation_data;
tea-cv_violation_data = cv_violation_data;

5.4.7 Process Roadside Fare Collection Violations

Input Flows

enforcement_data_for_roadside_fare_collection
fare_collection_roadside_violation_information

Output Flows

bad_transit_roadside_fare_payment
enforcement_data_for_roadside_fare_collection
tea_fare_collection_roadside_violation_data

Description:

Overview: This process shall manage the details of fare collection violations reported by the Manage Transit function that have taken place at the roadside, i.e., at a transit stop. The process shall use the parameters in the store of roadside fare collection violation (enforcement) data to process and send the information to the correct law enforcement agency. This process shall also maintain the roadside fare collection enforcement data store, entering all information received from other processes.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'fare_collection_roadside_violation_information'.

Solicited Input Processing: This process shall receive the following data flow as a result of requests for data retrieval from the local data store:
(a) 'enforcement_data_for_roadside_fare_collection'.

Solicited Output Processing: This process shall provide the following output flow as a result of the above input being received:
(a) 'tea-fare_collection_roadside_violation_data';
(b) 'enforcement_data_for_roadside_fare_collection'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the unsolicited input is received, use this data and the contents of the data store to generate the roadside fare collection violation data to be sent to the law enforcement agency, and enter the new data in the local data store;
(c) manage the data in the store of roadside fare collection violation (enforcement) data.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;

Output Flow Dynamics Assumptions:

tea-fare_collection_roadside_violation_data = fare_collection_roadside_violation_information;
bad_transit_roadside_fare_payment = 1;

5.5 Update Emergency Display Map Data

Input Flows

fmup_emergency_display_update
request_emergency_display_update

Output Flows

map_data_for_emergency_display
tmup_request_emergency_display_update

Description:

Overview: This process shall provide updates to the store of digitized map data used as the background for displays of incidents and emergencies produced by processes in the Manage Emergency Services function. The process shall obtain the new data from a specialist data supplier or some other appropriate data source, on receiving an update request from the emergency system operator interface process within the Manage Emergency Services function.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flow:
(a) 'request_emergency_display_update'.

Solicited Input Processing: This process shall receive the following data flow as a result of output being sent to external functions:
(a) 'fmup-emergency_display_update'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmup-request_emergency_display_update';
(b) 'map_data_for_emergency_display'.

Functional Requirements: This process shall meet the following requirements:
(a) continuously monitor for the receipt of the unsolicited data flow shown above;
(b) when the data flow in (a) is received, generate the request for emergency display update from the map update provider (mup) and continuously monitor for receipt of the solicited input data flow;
(c) when the flow in (b) is received, output this data to the map data for emergency display data store;
(d) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of digitized map data;
(e) manage the data in the store of digitized map data.

User Service Requirements:

USR = 5.0;
USR = 5.1;
USR = 5.2;

Output Flow Dynamics Assumptions:

tmup-request_emergency_display_update = request_emergency_display_update;
map_data_for_emergency_display = fmup-emergency_display_update;

5.6 Manage Emergency Services Data

Input Flows

archive_manage_emergency_vehicle_data
archive_provide_emergency_service_allocation_data
em_archive_request
em_archive_status
emergency_data_archive
emergency_service_log_for_archive
feso_archive_commands

Output Flows

em_archive_data
emergency_data_archive
teso_archive_status

Description:

Overview: This process shall collect emergency service data, emergency vehicle management data, emergency vehicle data, and incident data. It shall distribute this data to the Manage Archive Data Request where it can be archived and accessed upon request or upon receipt of fresh data.

All inputs to this process are unsolicited, and all outputs are solicited, except that the 'em_archive_status' is a solicited input.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited data inputs shown above is received, the process shall store them in the data store along with meta data (data attributes about the data), and update the catalog;
(c) when the unsolicited input from the emergency system operator is received, the process shall update the data store accordingly;
(d) when the request for emergency archive data is received, the process shall immediately generate the solicited output shown above from the data store;
(e) the process should then receive the emergency archive status solicited input and send this status to the emergency system operator;
(f) data shall only be sent to the source from which the data request originated;
(g) before output, the process shall put the data into a format that is easily read and interpreted by external processes and can also be read by travelers and users with the minimum of further processing.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.5;
USR = 7.1.3.1.5(a);
USR = 7.1.3.1.5(b);
USR = 7.1.3.1.5(c);
USR = 7.1.3.1.5(d);
USR = 7.1.3.1.5(f);
USR = 7.1.3.1.5(h);

Output Flow Dynamics Assumptions:

em_archive_data = em_archive_request;
teso-archive_status = em_archive_status;

6.1.1 Provide Trip Planning Information to Traveler

Input Flows

current_conditions
multimodal_service_data_response
paratransit_personal_schedule
paratransit_route_request
parking_lot_availability
prices
rideshare_response
supplied_route
traveler_current_condition_request
traveler_personal_current_condition_request
traveler_personal_trip_request
traveler_trip_request
trip_planning_parameters

Output Flows

multimodal_service_data_request
paratransit_route_response
paratransit_trip_request
parking_lot_data_for_archive
parking_lot_data_request
request_prices
traveler_personal_trip_information
traveler_rideshare_request
traveler_trip_and_cond_requests_for_archive
traveler_trip_information
trip_information
trip_request

Description:

Overview: This process shall obtain all the information needed to fulfill the traveler's request for a trip. The process shall support the request for trips that require the use of one or more modes of transport, and shall use the preferences and constraints specified by the traveler in the trip request, plus data from the store of trip planning parameters, to select the most appropriate modes. It shall send details of the trip requirements to the specialized processes that provide route information for the different modes of transport. When route data is received back from these processes, this process shall ensure that the whole trip is covered by one coherent route for which all the data such as costs, arrival times, and modal change points are known. The information provided to the traveler by the process shall be sufficient to enable the traveler to understand the routing, modes and cost of the trip. The trip information shall be stored for possible use in subsequent trip confirmation. The process also includes parking lot data. This data is used in transactions requiring electronic payment of parking lot services, as well as for a traveler making a parking lot reservation. This process shall exchange all input and output data from and to the traveler with the appropriate traveler interface process. The traveler shall send parking lot data, traveler trip requests, and traveler current condition requests to the archival process.

Data Flows: The traveler trip request input data flow is unsolicited. All output flows are solicited and will themselves generate input flows. The following input flows contain data requested from or written to data stores:
(a) 'trip_information', write only;
(b) 'trip_planning_parameters', read only.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the traveler trip request input flow listed above;
(b) when the input flow in (a) is are received, output data flows to the data archival process and other processes requesting various types of routes, rideshare information, demand responsive trip requests, according to the preferences and constraints in the traveler's trip request and the parameters governing trip selection contained in the read only data store identified above;
(c) when the data has been returned, construct the trip and ensure that there are no breaks, i.e. where mode changes are involved, each segment begins and ends at a valid modal interchange point;
(d) if any of the segments do not join up, change the preferences and constraints and repeat (b) until a correct match is produced;
(e) in parallel with (c) and (d) compute the total cost of the trip, including all tolls, parking lot charges, transit fares, and other costs;

(f) when all calculations are complete, store the trip information in the local store for use if the traveler decides to confirm and then send the trip data to the process that provides the traveler interface using the traveler trip information output flow defined above;
(g) when a traveler requests current conditions or parking lot data in the data store is updated, output this information to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.1;
USR = 1.1.1.1;
USR = 1.1.1.1.1;
USR = 1.1.1.1.2;
USR = 1.1.1.1.3;
USR = 1.1.1.1.4;
USR = 1.1.1.1.5;
USR = 1.1.1.1.6;
USR = 1.1.2;
USR = 1.1.2.1;
USR = 1.1.2.1.1;
USR = 1.1.2.1.2;
USR = 1.1.2.1.3;
USR = 1.1.2.1.4;
USR = 1.1.2.1.5;
USR = 1.1.2.1.6;
USR = 1.1.2.1.7;
USR = 1.1.2.1.8;
USR = 1.1.3;
USR = 1.1.3.1;
USR = 1.1.3.1.1;
USR = 1.1.3.1.2;
USR = 1.1.3.1.3;
USR = 1.1.3.1.4;
USR = 1.1.3.2;
USR = 1.1.3.3;
USR = 1.1.3.3.1;
USR = 1.1.3.3.2;
USR = 1.1.4;
USR = 1.1.4.1;
USR = 1.1.4.1.1;
USR = 1.1.4.1.2;
USR = 1.1.4.1.3;
USR = 1.4.0;
USR = 1.4.1;
USR = 1.4.1.3;
USR = 1.4.3;
USR = 1.4.3.1;
USR = 1.4.3.3;
USR = 1.4.3.3(a);
USR = 1.4.3.3(b);
USR = 1.4.3.3(c);
USR = 1.4.4.3(d);
USR = 1.4.4.3(e);
USR = 1.4.4.3(f);
USR = 1.5.0;
USR = 1.5.2;
USR = 1.5.2.2;
USR = 1.5.2.2(d);
USR = 1.5.2.2(e);
USR = 1.5.2.2(f);
USR = 1.5.2.2(g);
USR = 1.5.2.2(h);
USR = 1.6;
USR = 1.6.4;
USR = 1.6.4(b);
USR = 1.6.4(c);

USR = 1.7;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(f);
USR = 1.7.1.1.1(g);
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(d);
USR = 1.8.1.2(e);
USR = 1.8.1.3;
USR = 1.8.1.3(a);
USR = 1.8.1.3(e);
USR = 1.8.1.3(f);
USR = 1.8.1.6;
USR = 1.8.1.6(a);
USR = 1.8.2;
USR = 1.8.2.3;
USR = 1.8.2.3(c);
USR = 1.8.2.4;
USR = 1.8.2.4(a);
USR = 1.8.2.4(f);
USR = 2.0;
USR = 2.3.0;
USR = 2.3.1;
USR = 2.3.1.2;
USR = 2.4.0;
USR = 2.4.3;
USR = 2.4.3.1;
USR = 2.4.3.2;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(e);
USR = 7.1.3.1.8(g);

Output Flow Dynamics Assumptions:

paratransit_route_response = paratransit_route_request;
paratransit_trip_request = 5/(60*60)*ITS_PTRANSIT_TRAVS;
parking_lot_data_request = 5/(60*60)*ITS_RS_TRAVS;
request_prices = 20/(60*60)*ITS_TRAVS;
multimodal_service_data_request = 5/(60*60)*ITS_TRAVS;
traveler_personal_trip_information = traveler_personal_trip_request;
traveler_rideshare_request = 5/(60*60)*ITS_RS_TRAVS;
traveler_trip_information = 20/(60*60)*ITS_TRAVS;
trip_information = 20/(60*60)*ITS_TRAVS;
trip_request = 5/(60*60)*ITS_TRAVS;
parking_lot_data_for_archive = parking_lot_availability;
traveler_trip_and_cond_requests_for_archive = traveler_trip_request+traveler_personal_trip_request+
traveler_current_condition_request+traveler_personal_condition_request;

6.1.2 Confirm Traveler's Trip Plan

Input Flows

multimodal_service_confirmation
paratransit_route_confirm
parking_lot_reservation_confirm
rideshare_confirmation
traveler_payment_information
traveler_payment_response
traveler_personal_payment_information
traveler_personal_trip_confirmation
traveler_trip_confirmation
trip_information
trip_planning_parameters

Output Flows

multimodal_service_confirm
paratransit_service_confirmation
parking_lot_reservation_request
traveler_confirm_for_archive
traveler_payment_confirmation
traveler_payment_request
traveler_personal_payment_confirmation
traveler_personal_transaction_confirmation
traveler_rideshare_confirmation
traveler_transaction_confirmation

Description:

Overview: This process shall confirm a trip previously requested by a traveler and any financial transactions that this may require. The process shall base the trip confirmation upon information created by the process responsible for trip planning and stored locally. Confirmation details shall be sent to specialized processes (such as those responsible for demand responsive transit, ridesharing, etc.) to make reservations for their services. The response to these reservation requests and any necessary payment transactions shall be sent to the traveler. This process shall exchange all input and output data to and from the traveler via the appropriate traveler interface process. The trip confirmation shall be sent to the archival process.

Data Flows: The traveler trip confirmation input data flow is unsolicited. All output flows are solicited and will themselves generate input flows. The input flows 'trip_information' and 'trip_planning_parameters' contain data(implicitly) requested from data stores.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the traveler trip confirmation input flow listed above;
(b) when the input flow in (a) is are received, send the confirmation to the data archival process and obtain the data about the confirmed trip from the local trip information data store;
(c) output data flows to other processes requesting confirmation of various services and if required, advanced payment of tolls, parking lot charges, transit fares, and other costs;
(d) when all reservation and payment confirmations have been received, confirm the traveler's trip by sending the traveler trip confirmation information data flow identified above to the traveler interface process.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.3;
USR = 1.1.3.1;
USR = 1.1.3.2;
USR = 1.1.3.3;
USR = 1.1.4;
USR = 1.1.4.1;
USR = 1.1.4.1.1;
USR = 1.1.4.1.2;
USR = 1.1.4.1.3;
USR = 1.4.0;
USR = 1.4.1;
USR = 1.4.1.1;
USR = 1.4.1.2;
USR = 1.4.1.2(a);
USR = 1.4.1.3;

USR = 1.4.1.4;
USR = 1.4.2;
USR = 1.4.2.1;
USR = 1.4.2.4;
USR = 1.4.3;
USR = 1.4.3.2;
USR = 1.5.0;
USR = 1.5.1;
USR = 1.5.1.4;
USR = 1.8;
USR = 1.8.1;

Output Flow Dynamics Assumptions:

alternate_mode_service_confirm = (traveler_personal_trip_confirmation+traveler_trip_confirmation)/4;
multimodal_service_confirm = traveler_personal_trip_confirmation+traveler_trip_confirmation;
paratransit_service_confirmation = (traveler_personal_trip_confirmation+traveler_trip_confirmation)/10;
parking_lot_reservation_request = (traveler_personal_trip_confirmation+traveler_trip_confirmation)/8;
traveler_payment_confirmation = traveler_trip_confirmation;
traveler_payment_request = traveler_trip_confirmation;
traveler_personal_transaction_confirmation = traveler_personal_payment_information;
traveler_personal_payment_confirmation = traveler_personal_payment_information;
traveler_rideshare_confirmation = (traveler_personal_trip_confirmation+traveler_trip_confirmation)/10;
traveler_transaction_confirmation = traveler_trip_confirmation;
traveler_confirm_for_archive = traveler_personal_trip_confirmation_for_archive+
traveler_trip_confirmation_for_archive;

6.1.3 Manage Multimodal Service Provider Interface

Input Flows

fmtsp_air_services
fmtsp_ferry_services
fmtsp_multimodal_service_confirmation
fmtsp_rail_services
multimodal_service_confirm
multimodal_service_data_request
multimodal_service_details_data

Output Flows

multimodal_service_confirmation
multimodal_service_data_response
multimodal_service_details_data
tmtsp_air_services_request
tmtsp_confirm_multimodal_service
tmtsp_ferry_services_request
tmtsp_rail_services_request

Description:

Overview: This process shall collect data about services that are available to travelers from multimodal service providers. These suppliers shall be those that provide travel services that are not part of regular transit or demand responsive transit operations, e.g. heavy rail, and may not involve surface transportation, e.g. ferry and airline operations. The process shall provide data formatted for use as part of a traveler's multimodal trip, and shall support subsequent confirmation of any portion provided by the Multimodal Service Provider.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'multimodal_service_confirm';
(b) 'multimodal_service_data_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the multimodal transportation service provider:
(a) 'fmtsp-air_services';
(b) 'fmtsp-ferry_services';
(c) 'fmtsp-multimodal_service_confirmation';
(d) 'fmtsp-rail_services'.

Solicited Input/Output Processing: This process shall send and receive the following data flows as a means of reading data from and writing data to a data store:
(a) 'multimodal_service_details_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'multimodal_service_confirmation';
(b) 'multimodal_service_data_response';
(c) 'tmtsp-confirm_alternate_mode_service';
(d) 'tmtsp-ferry_services_request';
(e) 'tmtsp-rail_services_request';
(f) 'tmtsp-air_services_request'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the first input is received, request confirmation for the services that were specified;
(c) when a response to (b) is received, generate the first output flow identified above with the result of the confirmation request;
(d) when the second input is received, read data from the store of multimodal transportation service details to see if the required data has already been provided;
(e) if the data in (d) does not contain the required services or has been stored for a time that exceeds a locally determined threshold, generate the fourth to sixth solicited output data flows shown above that correspond to the modes in the input, including the required destination and arrival times;
(f) when the corresponding solicited inputs (first, second and fourth in the list shown above) are received, load the data into the data store;
(g) generate last solicited output data flow shown above using either the data read from the store or that provided by the multimodal transportation service provider.

User Service Requirements:

USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1.3;
USR = 2.2.1.1.4;

Output Flow Dynamics Assumptions:

multimodal_service_confirmation = multimodal_service_confirm;
multimodal_service_data_response = multimodal_service_data_request;
multimodal_service_details_data = multimodal_service_data_request;
tmtsp-confirm_multimodal_service = multimodal_service_confirm;
tmtsp-ferry_services_request = multimodal_service_data_request;
tmtsp-rail_services_request = multimodal_service_data_request;
tmtsp-air_services_request = multimodal_service_data_request;

6.1.4 Provide ISP Operator Interface for Trip Planning Parameters

Input Flows

fispo_trip_planning_parameters_request
fispo_trip_planning_parameters_update
trip_planning_parameters

Output Flows

tispo_trip_planning_parameters
trip_planning_parameters

Description:

Overview: This process shall manage the data store containing parameters used by the trip planning processes. These parameters shall govern the way in which multimodal trips are planned by other processes within Provide Trip Planning Services. This process shall accept inputs from the ISP Operator to define or update trip planning parameters. This process shall output these trip planning parameters to the ISP Operator.

(a) 'fispo-trip_planning_parameters_request':
(b) 'fispo-trip_planning_parameters_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'trip_planning_parameters'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tispo-trip_planning_parameters';
(b) 'trip_planning_parameters'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the first input is received, read the data from the store and generate the second output identified above;
(c) when the second input is received, update the contents of the store with the new data.

User Service Requirements:

USR = 1.6;
USR = 1.6.4;
USR = 1.6.4(b);
USR = 1.6.4(c);

Output Flow Dynamics Assumptions:

tispo-trip_planning_parameters = fispo-trip_planning_parameters_request;
trip_planning_parameters = fispo-trip_planning_parameters_update;

6.1.5 Collect Service Requests and Confirmation for Archive

Input Flows

advisory_data_request_for_archive
service_req_and_confirm_data
traffic_data_kiosk_request_for_archive
traffic_data_personal_request_for_archive
transit_deviation_kiosk_request_for_archive
transit_deviations_personal_request_for_archive
traveler_confirm_for_archive
traveler_route_accepted_for_archive
traveler_route_request_for_archive
traveler_trip_and_cond_requests_for_archive
traveler_yellow_pages_requests_for_archive
vehicle_guidance_route_accepted_for_archive
vehicle_route_request_for_archive
yellow_pages_advisory_requests_for_archive

Output Flows

service_req_and_confirm_data
service_req_and_confirm_for_archive

Description:

Overview: This process shall receive all traveler requests, such as requests for traffic and transit information, requests for current conditions such as weather, trip requests, guidance route requests, advisory requests, yellow page information requests, and service confirmations. These requests shall be stored in the service_req_and_confirm_data data store and output to the traveler information data archive. The process shall run when a new request or confirmation is received from an external source.

Data Flows: All input data flows are unsolicited. All output flows are solicited and will themselves generate input flows.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of any of the input flows listed above;
(b) when one of the input flows in (a) is received, write the data to the service_req_and_confirm_data store;
(c) output the data flow to the traveler information data archival process.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(g);
USR = 7.1.3.1.8(h);

Output Flow Dynamics Assumptions:

service_req_and_confirm_for_archive = 1;

6.1.6 Manage Traveler Info Archive Data

Input Flows

current_other_routes_use_for_archive
current_road_network_use_for_archive
fispo_archive_commands
parking_lot_data_for_archive
rideshare_data_for_archive
route_guidance_data_for_archive
service_req_and_confirm_for_archive
traveler_archive_request
traveler_archive_status
traveler_info_data_archive
traveler_info_payments_transactions
traveler_rideshare_request_for_archive
trip_request_for_archive
vehicle_guidance_probe_data_for_archive

Output Flows

tispo_archive_status
traveler_archive_data
traveler_info_data_archive

Description:

Overview: This process shall accept traveler information service requests and confirmations, parking management information, payment transaction data, rideshare requests, commercial and non-commercial probe data, route guidance data, and origin/destination data, and store it in its local traveler info data archive data store, together with a catalog to describe the data. When requested by the Manage Archive Data function, this information will be sent to that function. The process shall also provide a control interface to the ISP Operator, responding with the status received from the requester of the archive. The process shall run when a request for data or a catalog is received from an external source, when a command is received from the ISP Operator, or when fresh data is received.

Data Flows: All input data flows are unsolicited. All output flows are solicited
and will themselves generate input flows.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of any of the input flows listed above;
(b) when data to be archived is received, write the data to the traveler_info_data_archive and append its attributes;
(c) when a request for the archive catalog is received, read the traveler_info_data_archive for the catalog and output it to the Manage Archive function;
(d) when a request for the archive data is received, read the traveler_info_data_archive
and output the data requested to the Manage Archive function;
(e) when archive status is received, output the status to the ISP Operator;
(f) when a command is received from the ISP Operator, process the data in the traveler_info_data_archive as directed.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.4;
USR = 7.1.3.1.4(c);
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(a);
USR = 7.1.3.1.8(c);
USR = 7.1.3.1.8(d);
USR = 7.1.3.1.8(e);
USR = 7.1.3.1.8(f);
USR = 7.1.3.1.8(g);
USR = 7.1.3.1.8(h);

Output Flow Dynamics Assumptions:

tispo-archive_status = traveler_archive_status;
traveler_archive_data = traveler_archive_request;

6.2.1.1 Collect Traffic Data for Advisory Messages

Input Flows

foisp_traffic_data
foisp_traffic_information_request
planned_events
prediction_data
traffic_data_for_advisory_output
traveler_traffic_information_data_output

Output Flows

toisp_traffic_data_request
toisp_traffic_information
traffic_data_advisory_request
traveler_traffic_information_data_update

Description:

Overview: This process shall collect and fuse traffic data that will be used to create broadcast or advisory messages to travelers. The input data for this process shall consist of historical, current, and predicted traffic and planned event data. The process shall extract from the data those elements appropriate for advisory or broadcast messages and load it into the store of traveler_traffic_information_data store. The data can be provided to the process either via direct request from the process or as a result of periodic (unrequested) updates.

Data Flows: All input data flows are solicited with the exception of those listed above that contain prediction and planned events data. All output flows are solicited. Read and write access to the local store into which the input data is loaded after fusion is provided through the 'traveler_traffic_information_data' data flow.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above containing prediction data and planned events data;
(b) at locally determined times, generate the data output flows to processes in this and other functions, as listed above;
(c) collect the data returned as a result of (b) and load it with that received in (a) into the data store of traveler traffic information, fusing it with the data already present, deleting old data, e.g. that relating to incidents that are completed, etc.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.2;
USR = 1.2.2.1;
USR = 1.2.2.1.1;
USR = 1.2.2.1.2;
USR = 1.2.2.1.2.1;
USR = 1.2.2.1.3;
USR = 1.5;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(d);
USR = 1.5.2.5(g);
USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.1;
USR = 2.2.3;
USR = 2.2.3.1;
USR = 2.2.3.1.2;
USR = 2.2.3.1.2(a);
USR = 2.2.3.1.2(b);
USR = 2.2.3.2;
USR = 2.2.3.2.1;
USR = 2.2.3.2.2;

Output Flow Dynamics Assumptions:

traffic_data_advisory_request = 12/(60*60);
traveler_traffic_information_data = 12/(60*60);
toisp-traffic_information = foisp-traffic_information_request;
toisp-traffic_data_request = 12/(60*60);

6.2.1.2 Provide Traffic and Transit Advisory Messages

Input Flows

advisory_data_request
traveler_profile_from_vehicle
traveler_traffic_information_data
traveler_transit_information_data

Output Flows

advisory_data
advisory_data_request_for_archive

Description:

Overview: This process shall provide advisory data to users in vehicles (drivers or transit users) as a result of a request from the driver or transit user. (e.g. This process supports a request/response type of exchange with the user.) The advisory information is extracted from the data stores of traveler traffic and transit information. The process shall have the capability to filter the advisory data, read from the data stores, store so that the output only contains data that is relevant to the current location of the vehicle from which the request was made. When the user requests location specific data, the vehicle's location shall be provided to the process in the request message. Advisory data requests shall be sent to the data archival process.

Data Flows: The input data flow requesting advisory information is unsolicited. All output flows and the other input flows are solicited with the exception of the following:
(a) 'traveler_traffic_information_data' data flow - which contains data requested from a store;
(b) 'traveler_transit_information_data' data flow - which contains data requested from a store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flow requesting advisory data listed above;
(b) when the input in (a) is received, read the requested data from the store identified above, send the request to the data archival process, and generate the advisory data output flow identified above.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.2;
USR = 1.2.2.1;
USR = 1.2.2.1.1;
USR = 1.2.2.1.2;
USR = 1.2.2.1.2.1;
USR = 1.2.2.1.3;
USR = 1.5.0;
USR = 1.5.1;
USR = 1.5.1.1;
USR = 1.5.1.2;
USR = 1.5.1.2.1;
USR = 1.5.1.2.2;
USR = 1.5.1.2.3;
USR = 1.5.1.2.4;
USR = 1.5.1.2.5;
USR = 1.5.1.3;
USR = 1.5.1.4;
USR = 1.5.1.5;
USR = 1.5.2;
USR = 1.5.2.1;
USR = 1.5.2.2;
USR = 1.5.2.3;
USR = 1.5.2.4;
USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.1;
USR = 2.2.2;
USR = 2.2.2.1;
USR = 2.2.2.2;
USR = 2.2.2.3;

USR = 2.2.2.3(a);
USR = 2.2.2.3(b);

Output Flow Dynamics Assumptions:

advisory_data = 7*(fd-request_advisory_information)/10 + 7*(ftu-request_advisory_information)/10;
advisory_data_request_for_archive = advisory_data_request;

6.2.1.3 Collect Transit Data for Advisory Messages

Input Flows

foisp_transit_data
foisp_transit_information_request
transit_incident_data
transit_running_data_for_advisory_output
transit_services_for_advisory_data
traveler_transit_information_data_output

Output Flows

toisp_transit_data_request
toisp_transit_information
transit_conditions_advisories_request
transit_services_advisories_request
traveler_transit_information_data_update

Description:

Overview: This process shall collect and fuse transit advisory data that will be used to create broadcast or advisory messages to travelers. The process shall extract from the data those elements appropriate for advisory or broadcast messages and load it into the traveler_transit_information_data store. The data can be provided to the process either via direct request from the process or as a result of periodic (unrequested) updates. The process shall fuse all the received data into a coherent set, which is loaded into a traveler_transit_information_data store for access by other processes.

Data Flows: All input data flows are solicited as are all output flows. Read and write access to the local store into which the input data is loaded after fusion is provided through the 'traveler_transit_information_data' data flow.

Functional Requirements: This process shall:
(a) at locally determined intervals , generate the data output flows to processes in this and other functions, as listed above;
(b) collect the data returned as a result of (a) and load it into the local data store, fusing it with the data already present, deleting old data, for example that relate to incidents that are completed;

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.2;
USR = 1.2.2.1;
USR = 1.2.2.1.1;
USR = 1.2.2.1.2;
USR = 1.2.2.1.2.1;
USR = 1.2.2.1.3;
USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.1;
USR = 2.2.3;
USR = 2.2.3.1;
USR = 2.2.3.1.1;
USR = 2.2.3.1.2;
USR = 2.2.3.2;
USR = 2.2.3.2.1;
USR = 2.2.3.2.2;

Output Flow Dynamics Assumptions:

transit_conditions_advisories_request = 12/(60*60);
transit_services_advisories_request = 12/(60*60);
traveler_transit_information_data_update = 12/(60*60);
toisp-transit_information = foisp-transit_information_request;
toisp-transit_data_request = 12/(60*60);

6.2.1.4 Provide Traffic and Transit Broadcast Messages

Input Flows

broadcast_parameters_data
traveler_traffic_information_data
traveler_transit_information_data

Output Flows

broadcast_data

Description:

Overview: This process shall extract advisory data from stores of traveler traffic and transit information at locally determined intervals and send it out to drivers or transit users in vehicles as wide area broadcast messages. The content and rate of these messages shall be based upon parameters from the broadcast_parameters_data store, which is managed by the ISP operator.

Data Flows: The input data flow requesting advisory information is unsolicited. All output flows and the other input flows are solicited with the exception of the following:
(a) 'traveler_traffic_information_data' data flow - which contains data requested from a store;
(b) 'traveler_transit_information_data' data flow - which contains data requested from a store.

Functional Requirements: This process shall:
(a) at locally determined intervals read the data from the stores identified above and generate the broadcast data
output flow identified above;
(b) the data flow in (a) shall be generated using the filter parameters set up by the ISP operator and retained in a local data store.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.2;
USR = 1.2.2.1;
USR = 1.2.2.1.1;
USR = 1.2.2.1.2;
USR = 1.2.2.1.2.1;
USR = 1.2.2.1.3;
USR = 1.5.0;
USR = 1.5.1;
USR = 1.5.1.1;
USR = 1.5.1.2;
USR = 1.5.1.2.1;
USR = 1.5.1.2.2;
USR = 1.5.1.2.3;
USR = 1.5.1.2.4;
USR = 1.5.1.2.5;
USR = 1.5.1.3;
USR = 1.5.1.4;
USR = 1.5.1.5;
USR = 1.5.2;
USR = 1.5.2.1;
USR = 1.5.2.2;
USR = 1.5.2.3;
USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.1;
USR = 2.2.2;
USR = 2.2.2.1;

Output Flow Dynamics Assumptions:

broadcast_data = 12/(60*60);

6.2.1.5 Provide ISP Operator Broadcast Parameters Interface

Input Flows

broadcast_parameters_data_output
fispo_broadcast_data_parameters_request
fispo_broadcast_data_parameters_update

Output Flows

USR = 2.2.2.2;
broadcast_parameters_data_update USR = 2.2.2.3;
USR = 2.2.3;
tispo_broadcast_data_parameters_output USR = 2.2.3.1;

Description: USR = 2.2.3.1.1;
USR = 2.2.3.1.2;
Overview: This process shall provide the interface through which the ISP operator can manipulate data in the broadcast_parameters_data store. The data, in this store, shall be used by another process to define the scope and rate of wide area broadcast messages to vehicles. The process shall provide the ISP operator with the ability to request parameter data output and/or update the data store with new parameter values.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fispo-broadcast_data_parameters_request';
(b) 'fispo-broadcast_data_parameters_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores:
(a) 'broadcast_parameters_data_output'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'broadcast_parameters_data_update';
(b) 'tispo-broadcast_data_parameters_output'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the first input is received, read the data from the store of parameters using the solicited input flow identified above and generate the second solicited output flow identified above;
(c) when the second input is received, update the data in the store of parameters using the first solicited output flow shown above.

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

tispo-broadcast_data_parameters_output = fispo-broadcast_data_parameters_request;
broadcast_parameters_data_update = fispo-broadcast_data_parameters_update;

6.2.1.6 Provide Transit Advisory Data On Vehicle

Input Flows

ftu_destination_on_vehicle
ftu_other_services_vehicle_request
other_services_vehicle_response
transit_advisory_data_request
transit_vehicle_location
traveler_information_request
traveler_transit_information_data

Output Flows

other_services_vehicle_request
transit_advisory_data
transit_advisory_vehicle_information
traveler_information
ttu_other_services_vehicle_confirmed

Description:

Overview: This process shall gather transit advisory data and provide it via another process to the transit user on-board a transit vehicle. The interface shall receive requests from the transit user specifying the required destination of a transit service ride and other (yellow pages) type services. The transit user may also request and receive information about the state of traffic on the roadway, as well as transit route and stop data (i.e., traffic and transit advisory data). This process extracts data from the store of traveler transit information upon request for advisory data from the driver or transit user in a vehicle. The process shall filter the data read from the store so that output only contains that which is relevant to the current location of the vehicle from which the request was made. The vehicle's location shall be provided to the process in the request data. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited except for the following:

Unsolicited output:
(a) other_services_vehicle_request.

Solicited input:
(a) other_services_vehicle_response.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs from the transit user are received, generate the appropriate outputs identified above, prompting the user for any information that has not been supplied;
(c) when any response flow is received, generate the appropriate output to the transit user to indicate the success or failure of the requested transaction;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the transit user's payment information being transmitted, using any form of digital or analog encryption techniques.

User Service Requirements:

USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.6;
USR = 1.8.1.6(b);
USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.3;
USR = 2.2.1.2.2;
USR = 2.2.1.2.2.1;
USR = 2.2.1.2.2.2;
USR = 2.2.1.2.2.3;
USR = 2.2.1.2.2.4;

Output Flow Dynamics Assumptions:

other_services_vehicle_request = 12/(60*60)*ITS_TRANSIT_VEHS;
ttu-other_services_vehicle_confirmed = 12/(60*60)*ITS_TRANSIT_VEHS;

transit_advisory_data = 1*ITS_TRANSIT_VEHS;
traveler_information = 1*ITS_TRANSIT_VEHS;
transit_advisory_vehicle_information = 1*ITS_TRANSIT_VEHS;

6.2.2 Prepare and Output In-vehicle Displays

Input Flows

advisory_data
broadcast_data
driver_advisory_information_request
emergency_message_auto_output
emergency_message_driver_output
position_warnings
safety_warnings
transit_user_advisory_information_request
vehicle_control_status
vehicle_location_for_advisories
vehicle_signage_data
vehicle_smart_probe_data_output
vehicle_status_details_for_broadcast
vision_data
yellow_pages_advisory_data

Output Flows

advisory_data_request
driver_advisory_information
driver_broadcast_information
transit_user_advisory_information
traveler_profile_from_vehicle
yellow_pages_advisory_requests

Description:

Overview: This process shall provide in-vehicle advisory and broadcast data for output to drivers and transit users. The process shall format requests from users for advisory data and output the requests to other processes. The request for advisory data shall allow the user to request only information relevant to the location of the vehicle. The request may be repeated, periodically, or when the vehicle changes location by a distance determined by the implementation. Data broadcast to the driver shall include traffic related data(incidents, link data and in-vehicle signage), as well as data from the vehicle itself. This vehicle data includes vehicle conditions, smart probe data, safety and position warnings, and enhanced vision images. Safety and warning messages shall be prioritized by the process to supersede advisory and broadcast messages. The process shall also support the transfer of reservation requests from the users in vehicles for other services such as yellow pages.

Data Flows: All input data flows are unsolicited with the exception of that requesting input of advisory data. All output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs other than those from the driver, transit user, or location data are received, generate the output of advisory or broadcast data to the driver identified above;
(c) when the input from either the driver or the transit user is received, add the current vehicle location and send it as the advisory data request to another process;
(d) the data returned as a result of (c) shall be output to the driver or transit user depending on the source of the request for advisory information;
(e) when the broadcast data input is received, it shall be filtered to remove that which is not relevant to the wide area surrounding the vehicle before being output to the driver;
(f) repeat (c) and (d) for as long as the input is present, every time the vehicle location changes by an implementation distance.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.1;
USR = 1.2.1.1;
USR = 1.2.1.3;
USR = 1.2.1.5;
USR = 1.2.3;
USR = 1.2.3.1;
USR = 1.2.3.1.1;
USR = 1.2.3.1.2;
USR = 1.2.3.1.3;

USR = 1.2.3.1.4;
USR = 1.2.3.1.4.1;
USR = 1.2.3.1.4.2;
USR = 1.2.3.1.5;
USR = 1.2.3.2;
USR = 1.2.3.2.2;
USR = 1.2.3.2.2.1;
USR = 1.2.3.2.4;
USR = 1.2.3.2.5;

Output Flow Dynamics Assumptions:

advisory_data_request = fd-request_advisory_information+ftu-request_advisory_information;
driver_advisory_information = fd-request_advisory_information;
driver_broadcast_information = broadcast_data;
transit_user_advisory_information = ftu-request_advisory_information;
traveler_profile_from_vehicle = 1/(60*60*24);
yellow_pages_advisory_requests = (fd-request_advisory_information+ftu-request_advisory_information)/5;

6.2.3 Provide Transit User Advisory Interface

Input Flows

ftu_request_advisory_information
transit_advisory_data
transit_user_advisory_information
transit_vehicle_advisory_eta
transit_vehicle_eta_for_advisory
traveler_information

Output Flows

transit_advisory_data_request
transit_user_advisory_information_request
traveler_information_request
ttu_advisory_information
ttu_traveler_information

Description:

Overview: This process shall provide a data input and output interface for a transit user on-board a transit vehicle. The process shall enable traffic and travel advisory information, plus yellow pages information to be requested and output to the transit user. When constructing the outputs the process shall use the data in the store of vehicle display definitions data. In addition to the traveler's request/ response for information, broadcast advisories about the imminent arrival of the transit vehicle at the next stop are also displayed for the transit user. The process shall handle all inputs and outputs in such a way that they do not impair the vehicle driver's ability to control the transit vehicle in a manner that is safe to both its occupants, to other road and freeway users, and to pedestrians. The input and output forms shall also include those that are suitable for travelers with physical disabilities.

Data Flows: The input data flow from the transit user is unsolicited. All output data flows are solicited as is the input data flow of traveler advisory information. The data flow 'vehicle_display_definitions_data'

provides output template data from a local data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow listed above that is a request from the transit user for advisory information;
(b) when the input in (a) is received, generate the output information request output data flow identified above;
(c) when the input data flow of advisory information is received in response to (b), generate the output to the transit user using the template data available from the local data store;
(d) continue with (b) and (c) for as long as the input from the transit user is present.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.1;
USR = 1.2.1.1;
USR = 1.2.1.2;
USR = 1.2.1.3;
USR = 1.2.1.5;
USR = 1.2.3;
USR = 1.2.3.1;
USR = 1.2.3.1.1;
USR = 1.2.3.1.2;
USR = 1.2.3.1.3;
USR = 1.2.3.1.4;

Output Flow Dynamics Assumptions:

transit_user_advisory_information_request = ftu-request_advisory_information;
ttu-advisory_information = ftu-request_advisory_information;
ttu-traveler_information = ftu-request_advisory_information;
transit_advisory_data_request = ftu-request_advisory_information;
traveler_information_request = ftu-request_advisory_information;

6.2.4 Collect Yellow Pages Data

Input Flows

yellow_pages_data
yellow_pages_output

Output Flows

yellow_pages_data_request
yellow_pages_update

Description:

Overview: This process shall collect and fuse data about (yellow pages) services in order to provide information to users in vehicles. The process shall fuse all the received yellow pages data into a coherent set and loaded into the yellow_pages_information_data store for access by processes in response to requests from users in vehicles.

Data Flows: All input and output data flows are solicited. Read and write access to the local store is provided through the following data flows:
(a) 'yellow_pages output', contains data being read from the store;
(b) 'yellow_pages_update', contains data being written to the store.

Functional Requirements: This process shall meet the following functional requirements:
(a) at locally determined intervals , generate the data output flow to request other (yellow pages) services data, as listed above;
(b) collect the data returned as a result of (a) and load it into the local data store, fusing it with the data already present.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.2;
USR = 1.2.2.1;
USR = 1.2.2.1.1;
USR = 1.2.2.1.2;
USR = 1.2.2.1.2.1;
USR = 1.2.2.1.3;
USR = 1.5.0;
USR = 1.5.2;
USR = 1.5.2.2;
USR = 1.5.2.2(b);
USR = 1.5.2.2(g);
USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.1;
USR = 2.2.3;
USR = 2.2.3.1;
USR = 2.2.3.1.1;
USR = 2.2.3.2;
USR = 2.2.3.2.1;
USR = 2.2.3.2.2;
USR = 2.2.3.2.2(a);
USR = 2.2.3.2.2(b);
USR = 2.2.3.2.2(c);

Output Flow Dynamics Assumptions:

yellow_pages_update = 12/(60*60);
yellow_pages_data_request = 12/(60*60);

6.2.5 Provide Driver Interface

Input Flows

driver_advisory_information
driver_broadcast_information
fd_activate_vehicle_control
fd_request_advisory_information
vehicle_display_definitions_data

Output Flows

driver_advisory_information_request
td_advisory_information
td_broadcast_information
vehicle_control_request
vehicle_display_definitions_data

Description:

Overview: This process shall provide a user interface for a driver through which traffic and travel advisory information can be obtained. The process shall enable traffic and travel advisory information to be requested and output to the driver, and shall also support the automatic output of wide area broadcast information (including in vehicle signage) to the driver. The process shall support output of safety and vision enhancement information to the user. When constructing all outputs the process shall use the vehicle_display_definitions_data store parameters. One purpose of the vehicle_display_definitions_data store is to provide a translation table for road sign and message templates used for in-vehicle display. Part of the input interface provided by the process shall enable the driver to invoke and cancel automatic control of the vehicle including the use of automated highway system (ahs) lanes. The process shall support inputs from the driver in manual or audio form, and shall provide its outputs in audible or visual forms. Visual output may be either in hardcopy, or as a display. Both types of output shall not impair the driver's ability to control the vehicle in a safe manner.

Data Flows: All input data flows are unsolicited. The output flow is solicited as is the input of driver advisory information when it is received in response to driver input. The data flow 'vehicle_display_definitions_data' provides output template data from a local data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow listed above that is a request from the driver for advisory information;
(b) when the input in (a) is received, generate the output information request output data flow identified above;
(c) when the input data flow of advisory information is received in response to (b), generate the output to the driver using the template data available from the local data store;
(d) continue with (b) and (c) for as long as the input from the driver is present:
(e) if the request from the driver is for a change in the mode of vehicle automatic control, send the data to another process in the Provide Vehicle Control and Monitoring function using the vehicle control request data flow;
(f) if the driver advisory information data flow is received as unsolicited input, output the data that it contains immediately.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.1.5;
USR = 1.2.3;
USR = 1.2.3.2;
USR = 1.2.3.2.2;
USR = 1.2.3.2.2.1;
USR = 1.2.3.2.5;
USR = 1.3.0;
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.5;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(e);
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.6;
USR = 1.8.1.6(c);
USR = 6.5;
USR = 6.5.0;
USR = 6.5.3;
USR = 6.5.3.1;
USR = 6.5.3.1.2;

Output Flow Dynamics Assumptions:

driver_advisory_information_request = fd-request_advisory_information;
td-advisory_information = 1;
vehicle_control_request = fd-activate_vehicle_control;
td-broadcast_information = 1;

6.2.6 Provide Yellow Pages Data and Reservations

Input Flows

yellow_pages_advisory_requests
yellow_pages_information_data
yellow_pages_reservation_confirmation

Output Flows

yellow_pages_advisory_data
yellow_pages_advisory_requests_for_archive
yellow_pages_reservation_request

Description:

Overview: This process shall extract data from the yellow_pages_information_data store upon request for data from the driver or a transit user in a vehicle. The data read from the store may be filtered, by the process, so that output only contains that which is relevant to the current location of the vehicle. The process shall also enable the user to make reservations for yellow pages services from a vehicle. Yellow pages advisory requests shall be sent to the data archival process.

Data Flows: The input data flow requesting advisory information is unsolicited. All output flows and the other input flows are solicited with the exception of the following:
(a) 'yellow_pages_information_data' data flow - which contains data requested from a store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow requesting advisory data listed above;
(b) when the input in (a) is received, read the requested data from the store identified above, generate the advisory data output flow identified above, and output the advisory request to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.2;
USR = 1.2.2.1;
USR = 1.2.2.1.1;
USR = 1.2.2.1.2;
USR = 1.2.2.1.2.1;
USR = 1.2.2.1.3;
USR = 1.5.0;
USR = 1.5.1;
USR = 1.5.1.1;
USR = 1.5.1.2;
USR = 1.5.1.2.1;
USR = 1.5.1.2.2;
USR = 1.5.1.2.3;
USR = 1.5.1.2.4;
USR = 1.5.1.2.5;
USR = 1.5.1.3;
USR = 1.5.1.4;
USR = 1.5.1.5;
USR = 1.5.2;
USR = 1.5.2.1;
USR = 1.5.2.2;
USR = 1.5.2.2(a);
USR = 1.5.2.2(b);
USR = 1.5.2.2(h);
USR = 1.5.2.3;
USR = 1.5.2.3(a);
USR = 2.0;
USR = 2.2.0;
USR = 2.2.1;
USR = 2.2.1.1;

Output Flow Dynamics Assumptions:

yellow_pages_reservation_request = 3*(yellow_pages_advisory_requests)/10;
yellow_pages_advisory_data = 7*(yellow_pages_advisory_requests)/10;
yellow_pages_advisory_requests_for_archive = yellow_pages_advisory_requests;

6.3.1 Get Traveler Request

Input Flows

traveler_trip_planning_requests

Output Flows

traveler_current_condition_request
traveler_payment_information
traveler_traffic_condition_request
traveler_transaction_request
traveler_transit_condition_request
traveler_trip_confirmation
traveler_trip_request
traveler_yellow_pages_information_request

Description:

Overview: This process shall receive input data from a traveler located at a kiosk and send requests to the appropriate processes within the Provide Driver and Traveler Services function for further processing. The process shall provide support for trip planning, traffic, transit, yellow pages services information requests, trip confirmation, yellow pages confirmation, and payment requests. The actual interface to the traveler is provided through a separate process, which creates the input flow to this process.

Data Flows: The input data flow is unsolicited and all output flows are solicited.
Functional Requirements: This process shall:
(a) continuously monitor for receipt of the traveler trip planning input flow listed above;
(b) when the flow in (a) is received, extract the data and send it to the appropriate processes in the Provide Driver and Traveler Services function;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.3;
USR = 1.1.3.2;
USR = 1.1.3.2.1;
USR = 1.1.3.2.2;
USR = 1.1.3.2.3;
USR = 1.1.3.2.4;
USR = 1.1.3.2.5;
USR = 1.1.3.2.6;
USR = 1.1.3.2.7;
USR = 1.1.3.2.8;
USR = 1.1.3.2.9;
USR = 1.1.3.2.10;
USR = 1.4.0;
USR = 1.4.1;
USR = 1.4.1.1;
USR = 1.4.1.1(a);
USR = 1.4.1.1(b);
USR = 1.4.1.2;
USR = 1.4.1.2(b);
USR = 1.4.1.2(c);
USR = 1.4.1.2(d);
USR = 1.4.1.2(e);
USR = 1.4.1.3;
USR = 1.5.0;
USR = 1.5.2;
USR = 1.5.2.6;
USR = 1.5.2.6(b);

Output Flow Dynamics Assumptions:

traveler_current_condition_request = traveler_trip_planning_requests;
traveler_payment_information = (traveler_trip_planning_requests)/5;
traveler_traffic_condition_request = traveler_trip_planning_requests;
traveler_transaction_request = (traveler_trip_planning_requests)/5;
traveler_transit_condition_request = traveler_trip_planning_requests;
traveler_trip_confirmation = (traveler_trip_planning_requests)/5;
traveler_trip_request = traveler_trip_planning_requests;
traveler_yellow_pages_information_request = traveler_trip_planning_requests;

6.3.2 Inform Traveler

Input Flows

map_data_for_traveler_displays
traffic_data_for_broadcast_to_kiosks
traffic_data_for_kiosks
transit_deviations_for_broadcast_to_kiosks
transit_deviations_for_kiosks
transit_services_for_kiosks
traveler_payment_confirmation
traveler_traffic_condition_request
traveler_transaction_confirmation
traveler_transit_condition_request
traveler_trip_information
traveler_yellow_pages_data

Output Flows

traffic_data_kiosk_request
transit_deviation_kiosk_request
transit_services_kiosk_request
traveler_trip_planning_responses

Description:

Overview: This process provides the traveler (located at a kiosk) with data about all requested trip, traffic, transit, yellow pages services information, confirmation of any requested reservations, and payments made as part of confirmed trip plans. The data is sent by the process to an interface process that is responsible for its actual output to the traveler. This data may include digitized map data to act as the background to the output when the data is to be shown in a suitable format. This process shall receive data from other ITS functions by requesting it or be sent data as a result of requests from anotherprocess.

Data Flows: All output flows are solicited and all input data flows are unsolicited with the exception of the following:
(a) 'traffic_data_for_kiosks' - which is received as a result of output being sent to another process;
(b) 'transit_deviations_for_kiosks' - which is received as a result of output being sent to another process;
(c) 'transit_services_kiosk_request' - which is received as a result of output being sent to another process;
(d) 'map_data_for_traveler_displays' - which contains data requested from a data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above that are not details of transit services, traffic data and the display map data;
(b) when any of the flows in (a) are received, retrieve the relevant digitized display map data from the local store and send the combined data to the traveler interface process;
(c) when the flow received in (a) contains a request for transit or traffic data, send the request to the relevant process in the Manage Transit or Manage Traffic function;
(d) the input data received as a result of (c) shall be combined with the relevant digitized display map data from the local store and sent to the traveler interface process;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.1;
USR = 1.1.1.1;
USR = 1.1.1.1.1;
USR = 1.1.1.1.2;
USR = 1.1.1.1.3;
USR = 1.1.1.1.4;
USR = 1.1.1.1.5;
USR = 1.1.1.1.6;
USR = 1.1.2;
USR = 1.1.2.1;
USR = 1.1.2.1.1;
USR = 1.1.2.1.2;

Output Flow Dynamics Assumptions:

transit_deviation_kiosk_request = traveler_trip_planning_requests;
traffic_data_kiosk_request = traveler_trip_planning_requests;
transit_services_kiosk_request = traveler_trip_planning_requests;
traveler_trip_planning_responses = traveler_trip_planning_requests;

6.3.3 Provide Traveler Kiosk Interface

Input Flows

ft_extra_trip_data
ft_trip_planning_requests
traveler_regular_data
traveler_roadside_credit_identity
traveler_trip_planning_responses

Output Flows

traveler_regular_data
traveler_roadside_trip_costs
traveler_trip_planning_requests
tt_extra_trip_data_request
tt_trip_planning_responses

Description:

Overview: This process shall provide an interface at a kiosk through which travelers can input data and can receive data. The functions that the traveler can perform include plan and confirm trips, obtain current traffic and transit information, and declare emergencies. The process shall support the inclusion of yellow pages services such as lodging, restaurants, theaters, and other tourist activities as a part of trip planning and confirmation. The process shall be able to store frequently used data, such as the kiosk location, to reduce the amount of input needed by the traveler for each request. The process shall also carry out inputdata verification and require input confirmation before passing any of the traveler data to other processes (except when an emergency is being declared). The traveler's payment information shall be obtained by this process from another process specially designed for that purpose. The process shall support traveler inputs in manual or audio form, and shall provide its outputs in audible or visual forms consistent with a kiosk. These forms shall include those that are suitable for travelers with hearing or vision physical disabilities. The process shall enable viewing of data that has been previously output. Where it is appropriate, the process shall use the kiosk's location to filter data being displayed to only show information relevant to the kiosk'Æs location, or to a specific location requested by the user.

Data Flows: All input data flows are unsolicited and all output flows are solicited, with the exception of the 'traveler_regular_data' data flow which contains data requested from or written to a data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows from the traveler listed above;
(b) when any of the inputs in (a) are received, check for content and if necessary utilize data from the local store identified above;
(c) generate the output identified above and load the requested data into the local data store;
(d) continually monitor the data in the local store and compare it with that being input by travelers, deleting any data from the store which is not frequently used;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.3;
USR = 1.1.3.2;
USR = 1.1.3.2.1;
USR = 1.1.3.2.2;
USR = 1.1.3.2.3;
USR = 1.1.3.2.4;
USR = 1.1.3.2.5;

Output Flow Dynamics Assumptions:

traveler_regular_data = (ft-trip_planning_requests)/20;
traveler_roadside_trip_costs = ft-trip_planning_requests;
traveler_trip_planning_requests = ft-trip_planning_requests;
tt-extra_trip_data_request = ft-extra_trip_data;
tt-trip_planning_responses = ft-trip_planning_requests;

6.3.4 Update Traveler Display Map Data at Kiosk

Input Flows

fmup_traveler_display_update

Output Flows

map_data_for_traveler_displays
tmup_request_traveler_display_update

Description:

Overview: This process shall provide updates to the digitized map data used as the background for displays of trip, traffic and transit information. This data shall be suitable for use in kiosk displays. The process shall obtain the new data from map data suppliers or some other appropriate data source.

Unsolicited Output Processing: This process shall provide the following output flows regardless of any inputs being received:
(a) 'tmup-request_traveler_display_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to external functions:
(a) 'fmup-traveler_display_update'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above solicited input being received:
(a) 'map_data_for_traveler_displays'.

Functional Requirements: This process shall:
(a) send out the request for new data from the specialized digital map data supplier at periodic intervals (e.g. once per month) automatically so as to provide an up to date map display using the unsolicited output data flow shown above;
(b) as a result of the output of the data flow in (a) continuously monitor for receipt of the solicited input data flow shown above;
(c) when the flow in (b) is received, output the second solicited output data flow shown above;
(d) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of digitized map data.

User Service Requirements:

USR = 1.1;
USR = 1.1.3;
USR = 1.1.3.1;
USR = 1.1.3.1.1;
USR = 1.1.3.2.8;
USR = 1.1.4;
USR = 1.1.4.1;
USR = 1.1.4.1.3;
USR = 1.5;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(f);

Output Flow Dynamics Assumptions:

tmup-request_traveler_display_update = 12/(60*60*24*7*52);
map_data_for_traveler_displays = 12/(60*60*24*7*52);

6.4.1 Screen Rider Requests

Input Flows

rideshare_eligibility_data
traveler_rideshare_request

Output Flows

rideshare_data_for_archive
rideshare_ineligible_status_notification
rideshare_request_from_eligible_traveler
traveler_rideshare_request_for_archive

Description:

Overview: This process shall accept and screen traveler requests for ride-sharing. These requests shall be sent to the process as a result of trip plan requests received from travelers by other processes. This process shall use eligibility data from a local rideshare_data store, to screen travelers before they are matched with other travelers and to enable ridesharing for all or part of their proposed trips. Traveler rideshare requests and rideshare data from the rideshare_data data store shall be sent to the data archival process.

Data Flows: Both output flows are solicited. The input data flow from the traveler is unsolicited, whilst the other input flow 'rideshare_eligibility_data' contains data read from the store of rideshare data.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow from the traveler listed above;
(b) when the input in (a) is received, check the traveler's eligibility for ridesharing against the data in the local rideshare data store and send the request and rideshare data to the data archival process;
(c) if the traveler is eligible for ridesharing, send the eligible ride share request data flow to another process for matching of the traveler's proposed trip with others;
(d) if the traveler in ineligible for ridesharing, send the ineligible status data flow to the process that reports the results of the rideshare application to the process from which the rideshare request was sent.

User Service Requirements:

USR = 1.0;
USR = 1.4.0;
USR = 1.4.1;
USR = 1.4.1.4;
USR = 1.4.2;
USR = 1.4.2.1;
USR = 1.4.2.2;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(d);
USR = 1.8.1.2(g);
USR = 1.8.1.3;
USR = 1.8.1.3(d);
USR = 1.8.1.3(g);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(d);
USR = 1.8.2.4;
USR = 1.8.2.4(d);
USR = 1.8.2.4(g);
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.4;
USR = 7.1.3.1.4(c);

Output Flow Dynamics Assumptions:

rideshare_request_from_eligible_traveler = 5/(60*60)*ITS_RS_TRAVS;
rideshare_ineligible_status_notification = 5/(60*60)*ITS_RS_TRAVS;
rideshare_data_for_archive = rideshare_eligibility_data;
traveler_rideshare_request_for_archive = traveler_rideshare_request;

6.4.2 Match Rider and Provider

Input Flows

rideshare_confirmation_data
rideshare_data
rideshare_request_from_eligible_traveler
traffic_data_for_ridesharing

Output Flows

rideshare_data
rideshare_selection
traffic_data_ridesharing_request

Description:

Overview: This process shall match travelers for ridesharing trips. The process shall attempt to achieve a match by considering some or all of the following: the origin and destination of the traveler's proposed trip, any routing constraints, preferences specified by the traveler, compatibility of this rideshare with rideshares confirmed by other travelers, the requesting traveler's eligibility data, and traffic data obtained on request from the Manage Traffic function. The process shall consider the possible disbenefits to other travelers who will be part of the same rideshare when finding the rideshare best suited to the travelerÆs requirements. Theprocess shall store data about selected rideshares in the rideshare_data store, and shall update the data when confirmation of the rideshare acceptance is received from another process.

Data Flows: The input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'rideshare_data', which contains data requested from or written to a data store;
(b) 'traffic_data_ridesharing_request', which contains a request for traffic data from the Manage Traffic function;
(c) 'traffic_data_for_ridesharing', which contains the response to the request for traffic data from the Manage Traffic function.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow containing the eligible request data, as identified in the list above;
(b) when the input in (a) is received, match the traveler's rideshare request by comparing its routing with data on existing rideshares in the local data store, and producing a new route for the requesting traveler and one or more existing rideshare travelers;
(c) check that the route computed in (b) against traffic data requested from the Manage Traffic function to ensure that there will be no significant penalties from predicted occupancy levels and travel times;
(d) when all the criteria for sharing a ride have been satisfied and other traveler's have been found whose trips will match all or part of that in the current request, send the trip details to another process for the reporting of the results to the process from which the traveler's request was received;
(e) when the input is received indicating that the rideshare has been confirmed, update the data relating to the rideshare in the store of rideshare data.

User Service Requirements:

USR = 1.0;
USR = 1.4.0;
USR = 1.4.1;
USR = 1.4.1.2;
USR = 1.4.1.3;
USR = 1.4.1.4;
USR = 1.4.3;
USR = 1.4.3.4;
USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.2;
USR = 1.8.1.2(d);
USR = 1.8.1.2(g);
USR = 1.8.1.3;
USR = 1.8.1.3(d);
USR = 1.8.1.3(g);
USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(d);
USR = 1.8.2.4;
USR = 1.8.2.4(d);

USR = 1.8.2.4(g);

Output Flow Dynamics Assumptions:

rideshare_selection = 5/(60*60)*ITS_RS_TRAVS;
rideshare_data = 5/(60*60)*ITS_RS_TRAVS;
traffic_data_ridesharing_request = 12/(60*60);

6.4.3 Report Ride Match Results to Requestor

Input Flows

rideshare_ineligible_status_notification
rideshare_selection
traveler_rideshare_request

Output Flows

rideshare_response

Description:

Overview: This process shall report ridesharing match results to requesters. The data for the results shall be provided to this process by other processes responsible for assessing traveler eligibility, and the actual match with travelers in other rideshares. The process shall output data indicating a failure when either the data from the eligibility process shows a failure, or no ridesharing match can be found. The process shall also determine that no ridesharing match can be found if no match is found between the traveler's rideshare request and the rideshare data provided as input to it by another process. When a successful match is found, the process shall output the rideshare data to the process from which the traveler's request was received.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'rideshare_ineligible_status_notification';
(b) 'rideshare_selection';
(c) 'traveler_rideshare_request'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'rideshare_response'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received from the screening process indicating a failure, output this in the rideshare response data flow;
(c) when the rideshare details input is received, check that the rideshare matches the traveler's rideshare request;
(d) if the test in (c) is true, generate the rideshare response with the details of the rideshare;
(e) if the test in (c) fails, generate the rideshare response with a failure indicated.

User Service Requirements:

USR = 1.0;
USR = 1.4.1;
USR = 1.4.1.3;
USR = 1.4.1.4;

Output Flow Dynamics Assumptions:

rideshare_response = 5/(60*60)*ITS_RS_TRAVS;

6.4.4 Confirm Traveler Rideshare Request

Input Flows

rideshare_payment_confirmation
traveler_rideshare_confirmation

Output Flows

rideshare_confirmation
rideshare_confirmation_data
rideshare_payment_request

Description:

Overview: This process shall confirm the traveler's rideshare match and initiate a payment transaction where appropriate. The process shall send the payment transaction data for action by a process in the Provide Electronic Payment Services function. The results of this transaction shall be sent by this process to the process providing the overall trip confirmation. Once a rideshare match is confirmed, this data is sent to the rideshare match process where it can be factored in to subsequent matches.

Data Flows: The input data flow of traveler confirmation is unsolicited. All other input and output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow confirming the traveler's rideshare request, as shown in the list above;
(b) when the input in (a) is received, generate the output to request payment for obtaining the rideshare, identified in the above list;
(c) when as a result of (b) the payment confirmation data flow is received, generate the rideshare confirmation data flow to the process that provided the rideshare confirmation and to the process that matches rider and provider as it also manages the store of rideshare data;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 2.0;
USR = 2.3.0;
USR = 2.3.1;
USR = 2.3.1.3;
USR = 2.3.1.4;

Output Flow Dynamics Assumptions:

rideshare_confirmation = 1/(60*60)*ITS_RS_TRAVS;
rideshare_confirmation_data = 1/(60*60)*ITS_RS_TRAVS;
rideshare_payment_request = 1/(60*60)*ITS_RS_TRAVS;

6.5.1 Collect and Update Traveler Information

Input Flows

fm_traveler_information
fws_current_weather
fws_predicted_weather
fypsp_yellow_pages_data
incident_information
tourist_information
yellow_pages_new_data_request
yellow_pages_service_provider_data
yellow_pages_update_request

Output Flows

current_conditions
incident_information_request
tm_traveler_information_request
tourist_information
typsp_yellow_pages_info_request
yellow_pages_update_response

Description:

Overview: This process shall collect and update data about incidents, road construction, weather, events and yellow pages data. This data shall be obtained by the process from other ITS functions and from outside sources such as the weather service, yellow pages service providers and the media. The process shall load the data into a local store for use by the process that provides yellow pages information and reservations.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fws-current_weather';
(b) 'fws-predicted_weather';
(c) 'yellow_pages_new_data_request';
(d) 'yellow_pages_update_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'incident_information';
(b) 'yellow_pages_service_provider_data', which contains data requested from a data store;
(c) 'fm-traveler_information';
(d) 'fypsp-yellow_pages_data';
(e) 'tourist_information', which contains data requested from a data store.

Unsolicited Output Processing: This process shall provide the following output flows regardless of any inputs that are received:
(a) 'incident_information_request';
(b) 'tm-traveler_information_request';
(c) 'typsp-yellow_pages_info_request'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tourist_information', which contains data written to a data store;
(b) 'typsp-yellow_pages_info_request';
(c) 'yellow_pages_update_response'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when either of the weather service inputs are received, load the data into the store of tourist information using the solicited output flow shown above;
(c) when either of the yellow pages data flows are received in (a) send the yellow pages data request shown above in the list of unsolicited output flows;
(d) when the response to (c) is received in the solicited yellow pages input flow, load the data into the store of tourist information using the solicited output flow shown above;
(e) before loading data into the store of tourist information, read the current data from the store and amalgamate it with the new data;
(f) be responsible for the management of the data in the store of tourist information, using the most appropriate mechanism(s) such as a relational database, for storing the data;
(g) use the most appropriate mechanism(s) such as relational database , to read data from the store of information

and service provider data identified above.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.2;
USR = 1.1.2.1;
USR = 1.1.2.1.1;
USR = 1.1.2.1.2;
USR = 1.1.2.1.3;
USR = 1.1.2.1.4;
USR = 1.1.2.1.5;
USR = 1.1.2.1.6;
USR = 1.1.2.1.8;
USR = 1.5.0;
USR = 1.5.1;
USR = 1.5.1.1;
USR = 1.5.1.2;
USR = 1.5.1.3;
USR = 1.7;
USR = 1.7.1;
USR = 1.7.1.1;
USR = 1.7.1.1.1;
USR = 1.7.1.1.1(e);

Output Flow Dynamics Assumptions:

current_conditions = 5/(60*60);
incident_information_request = 5/(60*60);
tourist_information = 5/(60*60);
tm-traveler_information_request = 5/(60*60);
typsp-yellow_pages_info_request = yellow_pages_update_request;
yellow_pages_update_response = yellow_pages_update_request;

6.5.2 Provide Traveler Yellow Pages Information and Reservations

Input Flows

fypsp_transaction_confirmation
traveler_other_services_payment_result
traveler_payment_information
traveler_personal_payment_information
traveler_personal_transaction_request
traveler_personal_yellow_pages_information_request
traveler_transaction_request
traveler_yellow_pages_information_request
yellow_pages_data_request
yellow_pages_reservation_request
yellow_pages_update_response

Output Flows

traveler_other_services_payment_request
traveler_personal_yellow_pages_data
traveler_yellow_pages_data
traveler_yellow_pages_requests_for_archive
typsp_transaction_request
yellow_pages_data
yellow_pages_reservation_confirmation
yellow_pages_update_request

Description:

Overview: This process shall provide information and reservation services obtained from yellow pages service providers. The process shall provide the information and reservation data so that it can easily form part of a traveler's information request or trip planning activities. The process shall be able to request additional yellow pages information if the process cannot find the required data in the tourist_information data store. The process shall send requests for payment to a process in the Provide Electronic Payment Services function for action, and shall send the response back to the process from which the payment request was received. The traveler's yellow pages requests shall be sent to the data archival process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'yellow_pages_reservation_request';
(b) 'traveler_payment_information';
(c) 'traveler_transaction_request';
(d) 'traveler_personal_payment_information';
(e) 'traveler_personal_transaction_request';
(f) 'traveler_personal_yellow_pages_information_request';
(g) 'traveler_yellow_pages_information_request';
(h) 'yellow_pages_data_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'fypsp-transaction_confirmation';
(b) 'traveler_other_services_payment_result';
(c) 'yellow_pages_update_response', which contains data requested from a data store.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'yellow_pages_reservation_confirmation';
(b) 'traveler_other_services_payment_request';
(c) 'traveler_personal_yellow_pages_data';
(d) 'traveler_yellow_pages_data';
(e) 'typsp-transaction_request';

(f) 'yellow_pages_update_request';
(g) 'yellow_pages_data';
(h) 'traveler_yellow_pages_requests_for_archive'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the inputs are received, generate the solicited outputs as described below;
(c) unsolicited inputs (a) and (b) together generate solicited outputs (a) and (e), which in turn generate solicited inputs (b) and (a), which then generate solicited output (b);
(d) unsolicited inputs (c) and (e) will each generate solicited outputs (a) and (f), which in turn generate solicited inputs (b) and (a), which then generate solicited output (b);
(e) unsolicited inputs (f) and (g) generate solicited input (c), then the solicited outputs (d) and (e) respectively;
(f) unsolicited input (h) generates solicited input (c), then the solicited output (g);
(g) if in (e) or (f) the required data is not in the solicited input, then generate solicited output (f) and repeat the read of the input data;
(h) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(i) unsolicited inputs (c), (e), (f), and (g) will each generate solicited output (h), which in turn will be sent to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.5.0;
USR = 1.5.1;
USR = 1.5.1.3;
USR = 1.5.1.5;
USR = 1.5.2;
USR = 1.5.2.2;
USR = 1.5.2.2(a);
USR = 1.5.2.2(b);
USR = 1.5.2.2(h);
USR = 1.5.2.3;
USR = 1.5.2.3(a);
USR = 1.5.2.3(b);
USR = 1.5.2.4;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(g);

Output Flow Dynamics Assumptions:

yellow_pages_reservation_confirmation = yellow_pages_reservation_request;
traveler_other_services_payment_request = 1/(60*60)*ITS_TRAVS;
traveler_personal_yellow_pages_data = traveler_personal_yellow_pages_information_request;
traveler_yellow_pages_data = 5/(60*60)*ITS_TRAVS;
typsp-transaction_request = 1/(60*60)*ITS_TRAVS;
yellow_pages_update_response = 12/(60*60)*ITS_TRANSIT_VEHS+12/(60*60)*ITS_TRAVS;
yellow_pages_update_request = 12/(60*60);
traveler_yellow_pages_requests_for_archive = traveler_personal_transaction_request +traveler_transaction_request+traveler_personal_yellow_pages_information_request +traveler_yellow_pages_information_request;
yellow_pages_data = 1/(60*60);

6.5.3 Register Yellow Pages Service Providers

Input Flows

fypsp_provider_profile_update
fypsp_request_provider_registration
yellow_pages_service_provider_registration_response

Output Flows

typsp_provider_update_confirm
yellow_pages_new_data_request
yellow_pages_service_provider_data
yellow_pages_service_provider_registration_request

Description:

Overview: This process shall register yellow pages service providers. The process shall accept requests for registration from the providers and shall pass the data to a process in the Provide Electronic Payment Services function for action. The process shall send the result of this action to the provider, and if successful, shall send a request for the process that manages the contents of the store of tourist information to request data from the provider. The details of the provider shall also be loaded into the store used by that process, so that data from the provider can readily be obtained in the future. This process shall perform updating of the yellow pages service provider details.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fypsp-request_provider_registration';
(b) 'fypsp-provider_profile_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'yellow_pages_service_provider_registration_response'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'yellow_pages_service_provider_data', which is data output to a store;
(b) 'yellow_pages_service_provider_registration_request';
(c) 'typsp-provider_update_confirm';
(d) 'yellow_pages_new_data_request'.

Functional Requirements: This process shall :
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input in (a) is received, generate the solicited output in (b) identified above;
(c) when as a result of (b), the solicited input flow is received, generate the solicited outputs identified above (a), (c) and (d);
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

yellow_pages_service_provider_data = fypsp-request_provider_registration;
yellow_pages_service_provider_registration_request = fypsp-request_provider_registration;
typsp-provider_registration_confirm = fypsp-request_provider_registration;
yellow_pages_new_data_request = fypsp-request_provider_registration;
typsp-provider_update_confirm = fypsp-provider_profile_update;

6.6.1 Provide Multimodal Route Selection

Input Flows

cf_route_request
cv_route_request
fws_current_weather
fws_predicted_weather
other_route
paratransit_route_response
planned_events
transit_route
traveler_route_accepted
traveler_route_request
trip_request
vehicle_route

Output Flows

cf_route
cv_route
get_other_route
get_transit_route
get_vehicle_route
paratransit_route_confirm
paratransit_route_request
supplied_route
traveler_guidance_route
traveler_route_accepted_for_archive
traveler_route_request_for_archive
trip_request_for_archive

Description:

Overview: This process shall manage the creation of multimodal routes (those with one or more modes in them) in response to travelerÆs trip or route requests. It shall support on-line route guidance for travelers using personal devices, route guidance for vehicles, selection of specialized vehicle based routes for other ITS functions, (such as Manage Emergency Services and Manage Commercial Vehicles), and selection of multimodal routes in response to trip planning requests from travelers. The multimodal routes provided by the process shall take account of the traveler's preferences and constraints. Constraints can include the access needs of those with disabilities. Preferences can include minimizing waiting time at modal interchange points, level of traveler security, or minimum cost. Trip requests, traveler route requests, and traveler route acceptances shall be sent to the data archival process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'cv_route_request';
(b) 'fws-current_weather';
(c) 'fws-predicted_weather';
(d) 'planned_events';
(e) 'traveler_route_request';
(f) 'trip_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'other_route';
(b) 'paratransit_route_response';
(c) 'transit_route';
(d) 'traveler_route_accepted';
(d) 'vehicle_route'.

Solicited Output Processing: All outputs are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) priority shall be given to the response to requests for emergency vehicle routes and shall take precedence over all other activities;
(c) the response to any input that is a route request shall be to pass on the request to the appropriate Route Selection facility, bearing in mind the originator of the route (type of vehicle, or traveler, trip plan or route request) plus the preferences and constraints specified in the inputdata;

(d) when an emergency vehicle route has been determined, data showing the links, and the intersections along the route, including expected arrival times shall be sent to the Manage Traffic function to enable the current traffic management strategy to be modified to give the emergency vehicles a traffic control preemption;
(e) when trip or traveler guidance requests are received, the process shall be capable of automatically making several route requests of the specialized route selection facilities until it has determined the best multimodal route, bearing in mind the preferences and constraints included in the original request;
(f) the use of multimodal routes shall be aimed at minimizing the use of the private car in so far
as this is allowed by the preferences and constraints in the trip or traveler guidance request;
(g) the requirements of (e) and (f) above shall be superseded by the need to keep waiting time at transfer points to a minimum and non-existent for such things as late night travel or other situations where personal security may be a problem;
(h) if the process finds that a route cannot be provided within the preferences and constraints specified in the input request, the process shall produce a response which shall contain details of where the route cannot be selected;
(i) trip_request, traveler_route_request, or traveler_route_accepted will each generate outputs to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.2;
USR = 1.3.2.1;
USR = 1.3.2.2;
USR = 1.3.2.2.2;
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.3.3.1(b);
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.3;
USR = 1.4.0;
USR = 1.4.3;
USR = 1.4.3.3;
USR = 1.4.4.3(f);
USR = 1.4.4.3(g);
USR = 5.0;
USR = 5.2.0;
USR = 5.2.2;
USR = 5.2.2.1;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(f);
USR = 7.1.3.1.8(g);
USR = 7.1.3.1.8(h);

Output Flow Dynamics Assumptions:

cf_route = cf_route_request;
cv_route = cv_route_request;
emergency_vehicle_route = emergency_vehicle_route_request;
get_other_route = 5/(60*60)*ITS_TRAVS+2/(60*60)*ITS_GUIDED_TRAVS;
get_transit_route = 5/(60*60)*ITS_TRAVS+2/(60*60)*ITS_GUIDED_TRAVS;
get_vehicle_route = 4/(60*60*24*7)*CVO_VEHS+1/(60*60*24)*CVO_DVR+5/(60*60)*ITS_TRAVS;
paratransit_route_confirm = 2/(60*60)*ITS_GUIDED_TRAVS;
paratransit_route_request = 2/(60*60)*ITS_GUIDED_TRAVS;
supplied_route = 5/(60*60)*ITS_TRAVS;
traveler_guidance_route = 2/(60*60)*ITS_GUIDED_TRAVS;
trip_request_for_archive = trip_request;
traveler_route_request_for_archive = traveler_route_request;
traveler_route_accepted_for_archive = traveler_route_accepted;

6.6.2.1 Calculate Vehicle Route

Input Flows

fws_current_weather
fws_predicted_weather
get_vehicle_route
map_data_for_route_selection
route_segment_details
route_segment_details_updated
route_selection_parameters
routes_for_vehicles_data
vehicle_guidance_route_accepted
vehicle_route_request

Output Flows

logged_special_vehicle_route
request_route_segment_data
route_guidance_data_for_archive
routes_for_vehicles_data
special_vehicle_priority_routing
vehicle_guidance_route
vehicle_guidance_route_accepted_for_archive
vehicle_route
vehicle_route_request_for_archive

Description:

Overview: This process shall calculate trip planning and real-time dynamic guidance routes for all types of vehicles. The route data provided by the process in response to requests from vehicles using infrastructure based in-vehicle guidance shall only contain data necessary for the vehicle to provide guidance (since the data is intended for use by an in-vehicle navigation unit). The route provided for trip planning purposes shall contain data in a form which can be presented to a user via display (or alternatively in audio form). The process shall select the route according to the data included in the route request. Data provided by the requesting process includes preferences and constraints. The process shall have the capability of using current and/or predicted conditions of the road network in route calculation. The process shall have the capability of including additional factors such as current or predicted weather in the calculation of route. If the process cannot find the data it needs in the route_segment_details_data store, it shall request the process responsible for providing route calculation data to obtain it from the appropriate source. The process shall have the capability of outputting routes for special priority vehicles to the Manage Traffic function so that signal preemption could be provided for the special priority vehicle. The process shall send details of routes for commercial vehicles with hazardous or unusual loads to the Manage Incidents function for monitoring

(as a potential, or a planned event). Route guidance data and vehicle guidance route requests and acceptances shall be sent to the data archival process.

(a) 'get_vehicle_route';
(b) 'vehicle_route_request';
(c) 'fws-current_weather';
(d) 'fws-predicted_weather'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'other_road_details';
(b) 'road_details';
(c) 'road_details_updated';
(d) 'routes_for_vehicles_data';
(e) 'route_selection_parameters';
(f) 'vehicle_guidance_route_accepted'.

Solicited Output Processing: This process shall provide the following output flows as a result of the processing required by receipt of the above inputs:
(a) 'logged_special_vehicle_route';
(b) 'special_vehicle_priority_routing';
(c) 'request_route_segment_data';
(d) 'routes_for_vehicles_data';
(e) 'vehicle_guidance_route';
(f) 'vehicle_route';
(g) 'route_guidance_data_for_archive';

(h) 'vehicle_route_request_for_archive';
(i) 'vehicle_guidance_route_accepted_for_archive'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) the data received from the Weather Service shall be loaded into a local data store for use during the route determination process;
(c) the route selection process shall be performed using all the data provided in the request, and take account of the data received from the Weather Service and that in the store of road details;
(d) the route selection process shall only be performed when a request is received, or when the road details have been updated - see unsolicited input flows above;
(e) if during the course of the route selection process it is found that data for certain links does not exist in the local store, then the process shall check the store of other road details, and if the data is still not found, set up a request to the Provide Vehicle Route Calculation Data process for that data to be obtained;
(f) when the route has been selected, the time at which vehicles will use each route segment shall be entered into the store of routes for vehicles data;
(g) a selected route shall not include segments for which the maximum number of allowed vehicles has been exceeded, as specified in the store of road details - note that this is particularly important for segments that are part of an Automatic Highway System (AHS);
(h) if the number of vehicles on route segments within a selected route fall below preset values in the store of road details, that on completion of the route selection, send details of the links involved and the times at which intersections will be reached to the Manage Traffic function so that a form of traffic control preemption can be produced, for any given vehicle or class of vehicles;
(i) the process shall be responsible for the maintenance of the store of routes for vehicles data, using the appropriate mechanism(s) such as a relational database, for storing the data, and shall delete data
relating to vehicles that have now used the segments on their routes from that store;
(j) if the route for a commercial vehicle is constrained by its load, either because of size or its contents, e.g. HAZMAT loads, send a list of the links and arrival times at intersections to the Manage Traffic function (Manage Incidents facility);
(k) the decision on whether a selected route shall contain guidance information shall be based on the source of the request, and the time at which the route is to be used, i.e. those that are not to be used in the immediate future shall not contain guidance data;
(l) unsolicited inputs vehicle_route_request and vehicle_guidance_route_accepted will each generate an

output to be sent to the data archival process;
(m) updates to the routes_for_vehicles_data store will generate the route_guidance_data_for_archive output
to be sent to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.1;
USR = 1.2.1.4;
USR = 1.2.1.4.1;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.2.1(a);
USR = 1.3.1.2.1(b);
USR = 1.3.1.3;
USR = 1.3.1.3(a);
USR = 1.3.1.3(b);
USR = 1.3.2;
USR = 1.3.2.2;
USR = 1.3.2.2.1;
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.3.3.2;
USR = 1.3.3.2(a);
USR = 1.3.3.2(b);
USR = 1.3.3.2.1;

USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.3;
USR = 1.3.4.3.1;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(d);
USR = 7.1.3.1.8(g);
USR = 7.1.3.1.8(h);

Output Flow Dynamics Assumptions:

logged_special_vehicle_route = 1/(60*60*24);
special_vehicle_priority_routing = 6/(60*60);
request_route_segment_data = 12/(60*60);
routes_for_vehicles_data = 4/(60*60*24*7)*CVO_VEHS+1/(60*60*24)*CVO_DVR+5/(60*60)*ITS_TRAVS;
vehicle_guidance_route = vehicle_route_request;
vehicle_route = 4/(60*60*24*7)*CVO_VEHS+1/(60*60*24)*CVO_DVR+5/(60*60)*ITS_TRAVS;
vehicle_route_request_for_archive = vehicle_route_request;
vehicle_guidance_route_accepted_for_archive = vehicle_guidance_route_accepted;
route_guidance_data_for_archive = routes_for_vehicles_data;

6.6.2.2 Provide Vehicle Route Calculation Data

Input Flows

current_highway_network_state
current_road_network_state
link_data_for_guidance
other_route_segment_data
planned_events
prediction_data
request_route_segment_data
route_segment_details
routes_for_vehicles_data
traffic_data_for_guidance
vehicle_probe_data_amalgamation

Output Flows

current_road_network_use
current_road_network_use_for_archive
link_and_queue_data
link_data_store
request_other_route_segment_data
route_segment_details
route_segment_details_updated
traffic_data_guidance_request

Description:

Overview: This process shall update the data stores containing information which is used by the another process to calculate vehicle routes. This process shall also provide data about links (speed or travel times), and queues to be broadcast to vehicles (to support autonomous guidance with dynamic link updates). The process shall fuse link and queue data received from Manage Traffic sources with probe data received from vehicles under infrastructure based route guidance, or with probe data obtained from other sources (such as from an electronic toll collection system). The process shall obtain route segment data as requested data or as data received at periodic intervals from other ITS functions. The process shall have the capability to request data about route segments outside its own area by sending a data request to the process that provides the interface with other ISP's. Link addresses, mapped to other ISPs, will be maintained by this process and stored in the link_data_store. Usage of current road networks shall be sent to the data archival process.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'current_highway_network_state';
(b) 'current_road_network_state';
(c) 'link_data_for_guidance';
(d) 'planned_events';
(e) 'prediction_data';
(f) 'request_route_segment_data';
(g) 'vehicle_probe_data_amalgamation'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'link_data_store';
(b) 'map_database';
(c) 'other_route_segment_data';
(d) 'route_segment_details';
(e) 'routes_for_vehicles_data';
(f) 'traffic_data_for_guidance'.

Unsolicited Output Processing: This process shall provide the following output flows regardless of any inputs that are received:
(a) 'current_road_network_use';
(b) 'link_and_queue_data';
(c) 'traffic_data_guidance_request';
(d) 'current_road_network_use_for_archive'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'link_data_store';
(b) 'route_segment_details';
(c) 'request_other_rooute_segment_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) periodically send the first and third unsolicited output data flows listed above to the Manage Traffic function;
(c) at a rate determined by the implementation broadcast the second unsolicited output data flow to the Provide Vehicle Guidance facility, i.e. the frequency will be independent of the number of vehicles;
(d) when actual road data is received, load it into the appropriate data store for use by other processes within the Select Vehicle Route facility;
(e) each update of the data in a store should be followed by the sending of the road details updated data flow to the process that calculates vehicle routes and to the data archival process;
(f) if a request for data for route segments outside those in the local area is received, the process shall determine the source of supply of that data using the contents of the store of link data and send a request to the process within the Select Vehicle Route facility that has links to other data suppliers;
(g) the process shall be responsible for the maintenance of the store of road details both for local data and that obtained from other data suppliers, plus the store of data showing which links are not local and the identity of the supplier holding the data, using the most appropriate mechanism(s)

User Service Requirements:

such as a relational database.
USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.3.3.2;
USR = 1.3.3.2.1;
USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.3;
USR = 1.3.4.3.1;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(c);

Output Flow Dynamics Assumptions:

current_road_network_use = 12/(60*60);
link_data_store = 12/(60*60);
link_and_queue_data = 12/(60*60);
request_other_route_segment_data = 1/(60*60)*(ITS_GUIDED_VEHS)/100;
route_segment_details = vehicle_probe_data_amalgamation;
route_segment_details_updated = vehicle_probe_data_amalgamation;
traffic_data_guidance_request = 12/(60*60);
current_road_network_use_for_archive = 12/(60*60);

6.6.2.3 Provide Route Segment Data for Other Areas

Input Flows

foisp_data_supply
foisp_request_data
link_data_store
request_other_route_segment_data
route_segment_details

Output Flows

other_route_segment_data
toisp_data_supply
toisp_request_data

Description:

Overview: This process shall obtain from another ISP current or predicted data for road links that are outside the area served by the local supplier. This area, which may be defined on a geographic or jurisdictional basis, is the portion of the transportation network on which the ISP maintains real time information. Identification of which ISP to contact is based upon a store that maps a link to the ISP which maintains real time information about this link. If there is no map to another ISP in the data store, then the process will return default or static data for the link(s). This process shall also respond to similar requests from other ISPs for real time data on links within the local database.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'foisp-request_data';
(b) 'request_other_data'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to a terminator and requests for data retrieval:
(a) 'foisp-data_supply';
(b) 'link_data_store';
(c) 'road_details'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'other_route_segement_data';
(b) 'toisp-data_supply';
(c) 'toisp-request_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) use the most appropriate communications protocols to handle the data traffic on the link to the terminator, bearing in mind the transmission medium plus the size and frequency of the data flow.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.2.1(c);
USR = 1.7.0;
USR = 1.7.4;

Output Flow Dynamics Assumptions:

other_route_segment_data = request_other_route_segment_data;
toisp-data_supply = request_other_route_segment_data;

6.6.2.4 Update Vehicle Route Selection Map Data

Input Flows

fmup_route_selection_map_data
request_route_selection_map_update

Output Flows

map_data_for_route_selection
tmup_request_route_selection_map_update

Description:

Overview: This process shall provide the interface to map update providers, or other appropriate data sources, through which updates of the digitized map data can be obtained. The process shall request new data from the provider on request from the ISP operator interface process. The data received from the supplier shall be loaded into a the map_data_for_route_selection data store by the process in such a way that it can be easily used by the route selection process in determining vehicle routes, trip plannning, and on-line vehicle guidance.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_route_selection_map_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to the map provider terminator:
(a) 'fmup-route_selection_map_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'map_data_for_route_selection';
(b) 'tmup-request_route_selection_map_update'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input data flow listed above;
(b) when the input is received, generate the second output data flow identified above;
(c) when the new data is received from the map update provider terminator, update the data store using the first output data flow identified above;
(d) be responsible for the management of the data in the store of map data, using the appropriate mechanism(s) such as a relational database, for storing the data.

toisp-request_data = request_other_route_segment_data;

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.2;
USR = 1.3.2.1;
USR = 1.3.2.1(a);

Output Flow Dynamics Assumptions:

map_data_for_route_selection = request_route_selection_map_update;
tmup-request_route_selection_map_update = request_route_selection_map_update;

6.6.2.5 Provide ISP Operator Route Parameters Interface

Input Flows

fispo_request_other_routes_selection_map_data_update
fispo_request_route_selection_map_data_update
fispo_route_selection_parameters_request
fispo_route_selection_parameters_update
route_selection_parameters

Output Flows

request_other_routes_map_update
request_route_selection_map_update
route_selection_parameters
tispo_route_selection_parameters

Description:

Overview: This process shall provide the interface through which the ISP operator can input and update route calculation parameters used by the Provide Driver and Traveler Services function. The process shall provide an interface through which the ISP operator can review and request update of map data. The operator shall be able to use the process to request digitized map updates from suppliers, request output of trip planning and route selection control parameters, or to update the control parameters in the route_selection_parameters data store. The process shall support inputs from the ISP operator in manual or audio form, and shall provide its outputs in audible or visual forms. It shall enable the visual output to be in hardcopy, and/or display.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fispo-request_other_routes_selection_map_data_update';
(b) 'fispo-request_route_selection_map_data_update';
(c) 'fispo-route_selection_parameters_request';
(d) 'fispo-route_selection_parameters_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'route_selection_parameters'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'request_route_selection_map_update';
(b) 'request_other_routes_map_update';
(c) 'tispo_route_selection_parameters'.

Functional Requirements: This process shall meet:
(a) continuously monitor for receipt of the unsolicited input data flows listed above;
(b) when either of the first two inputs are received, generate the appropriate of the first two output data flows identified above;
(c) when the third input is received, read the data from the route selection parameters data store and output it in the third output data flow shown above;
(d) when the fourth input is received, load the data into the store of route selection parameters;
(e) be responsible for the management of the data in the store of route selection parameters, using the appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 1.3;
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.3.3.1(a);
USR = 1.3.4;
USR = 1.3.4.1;
USR = 1.3.4.1(a);

Output Flow Dynamics Assumptions:

request_route_selection_map_update = fispo-request_route_selection_map_data_update;
request_other_routes_map_update = fispo-request_other_routes_selection_map_data_update;
route_selection_parameters = fispo-route_selection_parameters_update;
tispo_route_selection_parameters = fispo-route_selection_parameters_request;

6.6.2.6 Calculate Vehicle Probe Data for Guidance

Input Flows

vehicle_guidance_probe_data
vehicle_toll_probe_data

Output Flows

vehicle_guidance_probe_data_for_archive
vehicle_probe_data_amalgamation

Description:

Overview: This process shall calculate route segment travel times from vehicle probe data. The probe data shall be accepted by the process from a variety of sources including toll collection points and vehicles receiving on-line infrastructure based guidance. The process shall be responsible for combining the data obtained from these sources and producing one set of route segment travel times or route segment speeds.The process shall indicate route segments for which no data, or insufficient data, is available (this indication could be by setting the link time or speed to zero). Vehicle guidance probe data shall be sent to the data archival process.

Unsolicited Input Processing: This process shall receive the following input unsolicited data flows:
(a) 'vehicle_guidance_probe_data';
(b) 'vehicle_toll_probe_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'vehicle_probe_data_amalgamation';
(b) 'vehicle_guidance_probe_data_for_archive'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input data flows listed above;
(b) when either of the input data flows is received, recalculate the route segment travel time for which the data applies, using an appropriate smoothing factor to remove any sudden fluctuations;
(c) periodically send the amalgamated route segment travel times calculated in (b) to the process that provides vehicle route calculation data;
(d) when new vehicle guidance probe data is received, the output data will in turn be sent to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.2;
USR = 1.3.2.1;
USR = 1.3.2.1(b);
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.3.3.2;
USR = 1.3.3.2.1;
USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.3;
USR = 1.3.4.3.1;
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(a);
USR = 7.1.3.1.8(c);

Output Flow Dynamics Assumptions:

vehicle_probe_data_amalgamation = 12/(60*60);
vehicle_guidance_probe_data_for_archive = vehicle_guidance_probe_data;

6.6.3 Update Other Routes Selection Map Data

Input Flows

fmup_other_routes_map_data
request_other_routes_map_update

Output Flows

map_data_for_general_use
tmup_request_other_routes_map_update

Description:

Overview: This process shall provide the interface to a map update providers through which to obtain fresh updates of digitized map data used in identification of non-vehicle portions of routes. The process shall request new data from the provider on request from the ISP operator interface process. The data received from the supplier shall be loaded into the map_data_for_general_use data store by the process in such a way that it can be easily used by the route selection process in determining non-vehicle routes for use in on-line traveler guidance and trip planning.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_other_routes_map_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to external functions:
(a) 'fmup-other_routes_map_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'tmup-request_other_routes_map_update';
(b) 'map_data_for_general_use'.

Functional Requirements: This process shall:
(a) continuously monitor for the receipt of the unsolicited data flow shown above;
(b) when the data flow in (a) is received, generate the first solicited output data flow shown above and continuously monitor for receipt of the solicited input data flow shown above;
(c) when the flow in (b) is received, output the second solicited output data flow shown above;
(d) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of digitized map data;
(e) be responsible for the management of the data in the store of the pollution data log, using the appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.2.1(b);

Output Flow Dynamics Assumptions:

map_data_for_general_use = fmup-other_routes_map_data;
tmup-request_other_routes_map_update = request_other_routes_map_update;

6.6.4 Select Transit Route

Input Flows

get_transit_route
transit_mode_routes
transit_route_details
transit_running_data_for_guidance
transit_services_for_guidance

Output Flows

current_transit_routes_use
transit_conditions_guidance_request
transit_mode_routes
transit_route
transit_route_details
transit_services_guidance_request

Description:

Overview: This process shall determine routes that are based on regular transit services. Routes shall be provided by the process to travelers in response to trip planning and on-line personal guidance requests. The data provided by the process shall be different for the two types of requests since trip planning information will not need the detail that guidance data requires. The process shall base routes on the current state of the regular transit services using data obtained from processes in the Manage Transit function. It shall also respond to any preferences and constraints, such as those for travelers with special needs, that are specified in the route request. Data on the current use of transit routes in on-line guidance shall be provided by the process to the Manage Demand function to aid in demand management. This data on current use of the transit routes in on-line guidance is stored in the transit_mode_routes data store.

Data Flows: The input data flow requesting a route is unsolicited. All other input flows and all output flows are solicited with the exception of the following which contains data requested from and written to data stores:
(a) 'transit_mode_routes';
(b) 'transit_route_details'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow requesting a transit route shown in the list of input data flows above;
(b) when the input in (a) is received, select a route that meets the requirements of the data in the constraints and preferences supplied as part of the request, using the data in the store of transit route details;
(c) as a result of (b) send the route details back to the requesting process in the transit route output data flow identified above and load the route details into the store of transit mode routes;
(d) periodically obtain new transit services data from the manage Transit function and load it into the store of transit route details;
(e) when the estimated arrival time of the last segment of each route expires, delete the entire route from the store of transit mode route details;
(f) periodically send a copy of the store of transit mode route details to the Manage Demand facility of the Manage Traffic function in the current transit routes use data flow;
(g) be responsible for the management of the data in the stores of transit route details and transit mode route details, using the appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.2.1(b);
USR = 1.3.1.2.1(c);
USR = 1.3.1.3;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.1;
USR = 1.3.3.2.2;
USR = 1.3.4;
USR = 1.3.4.3;
USR = 1.4.0;

USR = 1.4.3;
USR = 1.4.3.3;

Output Flow Dynamics Assumptions:

current_transit_routes_use = 12/(60*60);
transit_conditions_guidance_request = 12/(60*60);
transit_mode_routes = 12/(60*60);
transit_route = 5/(60*60)*ITS_TRAVS+2/(60*60)*ITS_GUIDED_TRAVS;
transit_route_details = 12/(60*60);
transit_services_guidance_request = 12/(60*60);

6.6.5 Select Other Routes

Input Flows

get_other_route
map_data_for_general_use
other_modes_routes
planned_events

Output Flows

current_other_routes_use
current_other_routes_use_for_archive
other_modes_routes
other_route

Description:

Overview: This process shall determine routes, or portions of routes, not based on use of vehicles or regular transit services. Routes shall be provided by the process for travelers in response to trip planning, on-line personal guidance requests, and for data archival. Data provided by the process will be different for the two types of requests since the data for trip planning will not need the level of detail that guidance data requires. The process shall calculate its routes using digitized map data obtained and updated by another process. It shall make use of the alternative modes, (such as ferries, walking, cycling, etc.) that have been specified in the route request, and shall also take account of any preferences and constraints, (such as those for travelers with special needs). Data on current use of routes in on-line guidance shall be provided by the process to the Manage Demand function.

Data Flows: The input data flow requesting a route is unsolicited. All other input flows and all output flows are solicited with the exception of the following which contains data requested from and written to data stores:
(a) 'other_modes_routes', read and write;
(b) 'map_data_for_general_use', read only.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flow requesting an other modes route, shown in the list of input data flows above;
(b) when the input in (a) is received, select a route that meets the requirements specified in the constraints and preferences supplied as part of the request, using the data in store of map data;
(c) as a result of (b) send the route details back to the requesting process in the other route output data flow identified above and load the route details into the store of other mode routes;
(d) when the estimated arrival time of the last segment of each route expires, delete the entire route from the store of other mode route details;
(e) periodically send a copy of the store of other mode route details to the Manage Demand facility of the Manage Traffic function in the current other routes use data flow;
(f) use the appropriate mechanism(s) such as a relational database, to retrieve data from the store of digitized
map data identified above;
(g) be responsible for the management of the data in the store of other mode route details, using the appropriate mechanism(s) such as a relational database, for storing the data;
(h) send the other routes currently used to the data archival process.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.2.1(d).2;
USR = 1.3.1.2.1(d).3;
USR = 1.3.1.3;
USR = 1.3.1.3(c);
USR = 1.3.1.3(d);
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.1;
USR = 1.3.3.2.2;
USR = 1.3.4;
USR = 1.3.4.3;
USR = 1.3.4.3.1;

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(c);

Output Flow Dynamics Assumptions:

other_modes_routes = 12/(60*60);
current_other_routes_use = 12/(60*60);
other_route = 5/(60*60)*ITS_TRAVS+2/(60*60)*ITS_GUIDED_TRAVS;
current_other_routes_use_for_archive = 12/(60*60);

6.7.1.1 Build Driver Personal Security Message

Input Flows

fd_emergency_request
vehicle_identity_for_driver_security_store
vehicle_location_for_emergencies
vehicle_status_details_for_driver_security

Output Flows

driver_personal_emergency_request

Description:

Overview: This process shall respond to the input of a request from a driver for action by the emergency services. Input of the request shall be received by the process from the driver via a panic button or some other functionally similar form of input device provided as part of the in-vehicle equipment. When the input is received, the process shall send a message to the communications process, containing the vehicle's current location, its identity and basic vehicle data relevant to its current condition, as well as any other data, such as personal medical history, vehicle orientation, etc., that may be developed in-vehicle by other systems.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'fd-emergency_request';
(b) 'driver_location_for_emergencies';
(c) 'vehicle_status_details_for_driver_security'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'vehicle_identity_store'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'driver_personal_emergency_request'.

Functional Requirements: This process shall:
(a) continuously monitor arrival of input data flow from the driver;
(b) when input from the driver is received obtain the current vehicle location and status and send this in a message with the vehicle identity and status to the communications process;
(c) if some or all of the data in (b) is missing, e.g. there is no current location, the message must be sent anyway, and repeated when a location becomes available.

User Service Requirements:

USR = 5.0;
USR = 5.1.0;
USR = 5.1.1;
USR = 5.1.1.1;
USR = 5.1.1.1(a);
USR = 5.1.1.1(b);
USR = 5.1.1.1(c);
USR = 5.1.1.1(d);
USR = 5.1.1.1(e);
USR = 5.1.1.2;

Output Flow Dynamics Assumptions:

driver_personal_emergency_request = 1;

6.7.1.2 Provide Driver In-vehicle Communications Function

Input Flows

driver_personal_emergency_request
emergency_request_driver_acknowledge

Output Flows

emergency_message_driver_output
emergency_request_driver_details

Description:

Overview: This process shall prepare and send an emergency message from a driver to the Manage Emergency Services function. The message shall only be sent by the process in response to data received from another process that monitors driver inputs. Once an emergency message has been sent, the process shall send a message to that effect to another process in the Provide Vehicle Monitoring and Control function for output to the driver. The process shall then await a response from the Manage Emergency Services function, and then send a detailed message to the other process for output to the driver. Output of the emergency message to the Manage Emergency Services function shall be repeated by the process at regular intervals until a response is received.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'driver_personal_emergency_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'emergency_request_driver_acknowledge'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'emergency_request_driver_details';
(b) 'emergency_message_driver_output'.

Functional Requirements: This process shall:
(a) transmission of the output message must be as near to instantaneous as possible following the receipt of data from the driver security message preparation process;
(b) the current data and time must be added to the data received from the driver security message preparation process;
(c) when the acknowledgment message is received it should be sent immediately to the interface process for driver advisory data;
(d) transmission of the output message must be repeated until an acknowledgment message is received updating the date and time as they change;
(e) initially, the message sent for output to the driver must show that data has been sent, and only be changed when an acknowledgment is received.

User Service Requirements:

USR = 5.0;
USR = 5.1.0;
USR = 5.1.1;
USR = 5.1.1.3;

Output Flow Dynamics Assumptions:

emergency_message_driver_output = 1;
emergency_request_driver_details = 1/(60*60*24*7*52)*(ITS_PVT_VEHS);

6.7.2.1.1 Determine In-vehicle Guidance Method

Input Flows

autonomous_vehicle_guidance_data
driver_guidance_accepted
driver_guidance_data
driver_guidance_request
dynamic_vehicle_guidance_data
retained_vehicle_guidance_data

Output Flows

autonomous_vehicle_guidance_accepted
autonomous_vehicle_guidance_data_request
driver_input_request
driving_guidance_instructions
dynamic_vehicle_guidance_data_request
retained_vehicle_guidance_data
vehicle_guidance_route_accepted

Description:

Overview: This process shall act as the interface for guidance requests received from drivers in vehicles. The process shall select the best method for in-vehicle guidance based on data in the driver's request. Three general methods of route guidance are supported: 1) dynamic (infrastructure based guidance is provided to the vehicle unit), 2) dynamic autonomous (link and queue speed or travel times are obtained

from the infrastructure and used by the autonomous in vehicle unit), and autonomous (the in vehicle unit

uses only locally available data- there is no information provided by the infrastructure). When dynamic guidance is selected, the vehicle's travel time for each link shall be provided by the process back to
a central source of data. If the communications link to the central source fails in either of the modes that use it, the process shall automatically revert to the use of local data only. When the original mode was centralized guidance, the process shall use the last set of guidance data that was received, and if this is not
sufficient for the vehicle to reach the requested destination, automatically revert to autonomous guidance

using local data only.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'driver_guidance_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'autonomous_vehicle_guidance_data';
(b) 'driver_guidance_accepted';
(c) 'driver_guidance_data';
(d) 'dynamic_vehicle_guidance_data';
(e) 'retained_vehicle_guidance_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'autonomous_vehicle_guidance_accepted';
(b) 'autonomous_vehicle_guidance_data_request';
(c) 'driving_guidance_instructions';
(d) 'driver_input_request';
(e) 'dynamic_vehicle_guidance_data_request';
(f) 'retained_vehicle_guidance_data'
(g) 'vehicle_guidance_route_accepted'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) select the appropriate process for the guidance data request dependent on whether the data input by the driver specifies the use of dynamic guidance;
(c) if dynamic guidance is not available and was requested, then the autonomous guidance process must be used;
(d) if dynamic guidance is used but becomes unavailable then the autonomous guidance process must be used from the point at which the dynamic guidance was lost;

(e) if dynamic guidance having been lost is regained, the first dynamic guidance request must use the vehicle's current location as the origin for the route request;
(f) the process shall be responsible for the maintenance of the store of data used in guidance requests using the appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.1;
USR = 1.2.1.4;

Output Flow Dynamics Assumptions:

autonomous_vehicle_guidance_accepted = driver_guidance_accepted;
autonomous_vehicle_guidance_data_request = driver_guidance_request;
driver_input_request = 2/(60*60*24);
driving_guidance_instructions = 1/(60);
dynamic_vehicle_guidance_data_request = driver_guidance_request;
retained_vehicle_guidance_data = 1/(60*60*24*7);
vehicle_guidance_route_accepted = driver_guidance_accepted;

6.7.2.1.2 Provide Dynamic In-vehicle Guidance

Input Flows

ahs_route_request
dynamic_vehicle_guidance_data_request
vehicle_guidance_route
vehicle_location_for_dynamic_guidance

Output Flows

ahs_route
dynamic_vehicle_guidance_data
vehicle_guidance_probe_data
vehicle_route_request

Description:

Overview: This process shall enable dynamic vehicle route guidance data to be calculated. The process shall perform the same dynamic vehicle route guidance services for vehicles that are under automatic control using automatic highway system (ahs)lanes. When providing dynamic guidance, the process provides vehicle travel times as probe data to another process in the Provide Driver and Traveler Services function. The process shall base its guidance request on data input by the driver through another process, and on the vehicleÆs current location as provided by another process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'ahs_route_request';
(b) 'dynamic_vehicle_guidance_data_request';
(c) 'vehicle_location_for_dynamic_guidance'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'vehicle_guidance_route'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'ahs_route';
(b) 'dynamic_vehicle_guidance_data';
(c) 'vehicle_guidance_probe_data';
(d) 'vehicle_route_request'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the second input in (a) is received send the driver route request data flow;
(c) if no origin is specified in the guidance data request, then the current vehicle location data obtained from the unsolicited input flow shall be used as the origin of the flow in (b);
(d) when the solicited input flow is received, load the data into the flow of dynamic driver guidance data and sent it to the process that determines the driver guidance method;
(e) every time a route request is made compute the time since the last request and send it with the vehicle's current position in the flow of vehicle probe data;
(f) when the first unsolicited input flow is received, send a request for a route based on using automatic highway system (ahs) lanes instead of the request in (b);
(g) the route data resulting from (f) should be sent to the process requesting the ahs route and no guidance data shall be output as in (d) above.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.2;
USR = 1.3.2.2;
USR = 1.3.2.3;
USR = 1.3.2.3.1;
USR = 1.3.3.1;
USR = 1.3.3.2;
USR = 1.3.3.2.2;
USR = 1.3.3.3;

USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.2.2;
USR = 1.3.4.3;

Output Flow Dynamics Assumptions:

ahs_route = 2/(60*60*24)*ITS_GUIDED_VEHS;
dynamic_vehicle_guidance_data = 1/(60);
vehicle_guidance_probe_data = 15/(60*60)*ITS_GUIDED_VEHS;
vehicle_route_request = 2/(60*60*24)*ITS_GUIDED_VEHS;

6.7.2.1.3 Provide Autonomous In-vehicle Guidance

Input Flows

autonomous_vehicle_guidance_accepted
autonomous_vehicle_guidance_data_request
link_and_queue_data
vehicle_location_for_autonomous_guidance
vehicle_map_database

Output Flows

autonomous_vehicle_guidance_data

Description:

Overview: This process shall provide autonomous in-vehicle guidance. It shall calculate the route using data obtained from an in-vehicle navigable map database which can be supplemented with link queue and travel time data obtained from a central source, if specified by the driver and available. The process shall provide guidance in the form of actual driving instructions, e.g. turn left at the next intersection, take the right lane, etc. When link queue and travel time data are being used, the process shall provide guidance for the best route for current traffic conditions, within the preferences and constraints specified by the driver in the guidance request.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'autonomous_vehicle_guidance_data_request';
(b) 'link_and_queue_data';
(c) 'vehicle_location_for_autonomous_guidance'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval and data being sent to another process:
(a) 'vehicle_map_database';
(b) 'autonomous_vehicle_guidance_accepted'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'autonomous_vehicle_guidance_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input data flows listed above;
(b) when the first input data flow in (a) is received, determine the shortest route using the data in the store of navigable map data;
(c) when requested by the driver and available, use the link queue and journey time data to modify the route obtained from using the in-vehicle map database data to produce the best route for the current traffic conditions;
(d) use the vehicle location data in the second input data flow in (a) to determine the origin for the route unless the driver has specified an origin in the guidance request;
(e) when the route has been determined, read the associated guidance instructions from the store and output them to the process requesting guidance.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.2;
USR = 1.3.2.2;
USR = 1.3.2.3;
USR = 1.3.2.3.1;
USR = 1.3.3.1;
USR = 1.3.3.2;
USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.2.2;
USR = 1.3.4.2.2(a);

USR = 1.3.4.3;

Output Flow Dynamics Assumptions:

autonomous_vehicle_guidance_data = 1/(60);

6.7.2.2 Process Vehicle Location Data

Input Flows

From_Location_Data_Source
vehicle_map_database

Output Flows

vehicle_location_for_advisories
vehicle_location_for_autonomous_guidance
vehicle_location_for_cv
vehicle_location_for_dynamic_guidance
vehicle_location_for_emergencies
vehicle_location_for_emergency_services
vehicle_location_for_incidents
vehicle_location_for_transit

Description:

Overview: This process shall provide the vehicle's current location. It shall calculate the location from one or more sources of position data such as GPS, DGPS, odometer and differential odometers, and shall refine its calculations using techniques such as map matching, etc. Location data (intended for use by in-vehicle navigation, guidance systems, and any emergency notification systems) should be provided by the process in a manner that is as precise as is practical within cost and technology constraints. Location data intended for transit vehicles and driver advisories may be less precise.

Data Flows: The input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above;
(b) continuously compute the vehicle's most probable current location using the data in the input flows, refinement and/or filtering algorithms (e.g. dead reckoning, map-following, etc).
(c) Provide the vehicle location to other processes in the Provide Driver and Traveler Services, Manage Commercial Vehicle, Manage Transit, and Manage Emergency Services functions using output flows as identified above;
(d) it shall be possible for the process to compute the location from as many sources of data as are simultaneously available to it, and to apply filtering and/or map matching algorithms as may be appropriate to consolidate or to choose among locations calculated from the various sources of data;
(e) vehicle location determination for transit and driver advisories may be of lesser precision than locations intended for navigation and route guidance processes.

User Service Requirements:

USR = 1.0;
USR = 1.2.0;
USR = 1.2.1;
USR = 1.2.1.1;
USR = 1.2.1.2;
USR = 1.2.1.3;
USR = 1.2.1.5;
USR = 1.2.3;
USR = 1.2.3.1;
USR = 1.2.3.1.1;
USR = 1.2.3.1.2;
USR = 1.2.3.1.3;
USR = 1.2.3.1.4;
USR = 1.2.3.1.4.1;
USR = 1.2.3.1.4.2;
USR = 1.2.3.1.5;
USR = 1.2.3.2;
USR = 1.2.3.2.2.1;
USR = 1.2.3.2.4;
USR = 1.2.3.2.5;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.2;
USR = 1.3.2.1;

USR = 1.3.2.2;
USR = 1.3.3;
USR = 1.3.3.1;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.2.2;
USR = 1.3.4.3;
USR = 2.2.1.2.2;
USR = 2.2.1.2.2.1;
USR = 2.2.1.2.2.2;
USR = 2.2.1.2.2.3;
USR = 2.2.1.2.2.4;
USR = 5.0;
USR = 5.1.0;
USR = 5.1.1;
USR = 5.1.1.1;
USR = 5.1.1.1(d);
USR = 5.1.1.2;
USR = 5.1.1.4;
; USR = 5.1.2;
USR = 5.1.2.1;
USR = 5.1.2.1.1;
USR = 5.1.2.1.2;
USR = 5.1.2.2
USR = 5.1.2.2(b);
USR = 6.0;
USR = 6.5.0;
USR = 6.5.1;
USR = 6.5.1.1;
USR = 6.5.1.1.1;
USR = 6.5.1.1.2;
USR = 6.5.1.1.3;
USR = 6.5.2;
USR = 6.5.2.1;
USR = 6.5.2.1.1;
USR = 6.5.2.1.2;
USR = 6.5.3;
USR = 6.5.3.1;
USR = 6.5.3.1.1;

Output Flow Dynamics Assumptions:

vehicle_location_for_autonomous_guidance = 1;
vehicle_location_for_cv = 1;
vehicle_location_for_emergencies = 1
vehicle_location_for_emergency_services = 1;
vehicle_location_for_dynamic_guidance = 1;
vehicle_location_for_incidents = 1;
vehicle_location_for_transit = 1;;
vehicle_location_for_advisories = 1;

6.7.2.3 Provide Driver Guidance Interface

Input Flows

driver_credit_identity
driver_input_request
driver_map_update_response
driving_guidance_instructions
fd_guidance_data
fd_guidance_map_update_request
fd_guidance_request
fd_guidance_route_accepted

Output Flows

driver_advanced_payment_for_map
driver_guidance_accepted
driver_guidance_data
driver_guidance_request
driver_map_update_request
td_driving_guidance
td_guidance_input_request
td_guidance_map_update_response
td_guidance_route_details

Description:

Overview: This process shall provide a user interface for the vehicle's driver through which route guidance is provided. Three types of route guidance provided by other processes shall be supported by this process(dynamic infrastructure based, autonomous with infrastructure data update, and autonomous). The process shall enable input by the driver of the type of guidance required, the data from which the route is to be determined and output of the resulting route. The process shall not provide on-line guidance until the route has been accepted by the driver. For those forms of guidance that require an on-board map database, the process shall provide an interface through which the driver may obtain and pay for an initial copy of the database plus updates when needed. The process shall support inputs from the driver in either manual or audio form, and shall provide its outputs in audible or visual forms. It shall enable the visual output to be either in hardcopy, and/or display. Both types of output shall not impair the driver's ability to control the vehicle in a safe manner.

Data Flows: All input data flows are unsolicited and all output flows are solicited

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows from the driver listed above;
(b) when any inputs are received, generate the appropriate outputs identified above;
(c) implementation of guidance will generate a succession of output data which must be passed on to the driver without the need for further input;
(d) the output in (c) must be retained until the next set of guidance data arrives for output;
(e) the advanced payment for map data flow is only generated when the driver credit identity data flow contains a stored credit value and not a credit identity;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the credit identity or stored credit being transmitted using any form of digital or analogue technique.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;

Output Flow Dynamics Assumptions:

driver_advanced_payment_for_map = fd-guidance_map_update_request;
driver_guidance_accepted = fd-guidance_route_accepted;
driver_guidance_data = 1/(60);
driver_guidance_request = fd-guidance_request;
driver_map_update_request = fd-guidance_map_update_request;
td-guidance_map_update_response = fd-guidance_map_update_request;
td-driving_guidance = 1/(60);
td-guidance_input_request = 30/(60*60);
td-guidance_route_details = fd-guidance_request;

6.7.2.4 Update Vehicle Navigable Map Database

Input Flows

driver_map_update_payment_response
driver_map_update_request
fmup_vehicle_map_update
fmup_vehicle_map_update_cost

Output Flows

driver_map_update_payment_request
driver_map_update_response
tmup_vehicle_map_update_cost_request
tmup_vehicle_map_update_request
vehicle_map_database

Description:

Overview: This process shall update the vehicle's navigable database based on digitized data obtained from a map provider, or other appropriate data source. The update shall be initiated by the driver through another process .The process shall have the capability to allow a financial transaction (to pay for the update) to be successfully completed using processes in the Provide Electronic Payment Services function. When the new map data is received, it shall be loaded by the process into the vehicle_map_database data store for use by other processes. The result of the update request (successful or not) shall be sent back to the driver interface process for output to the driver.

Data Flows: The driver update request input data flow is unsolicited. All other input flows and the output flows are solicited with the exception of the following which contains the new navigable map data to be written to its data store: 'vehicle_map_database'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the driver update request input flow listed above;
(b) when the input in (a) is received, send the data flow to the map update provider requesting the cost of the update;
(c) when a response to (b) is received and the credit identity not the stored credit was provided as part of the data in (a), generate the payment request;
(d) when a successful response is received to (c), generate the request to the map supplier for a new navigable map database;
(e) when a response to (b) is received and the stored credit not the credit identity was provided as part of the data in (a), compare it with the stored credit value and if greater send the update response data flow to the driver interface process showing an unsuccessful update;
(f) if the check in (e) shows that there is sufficient stored credit, generate the request to the map supplier for a new navigable map database;
(g) when the response to the map database requests in (d) or (f) is received, load the new navigable map data into the data store, and send the update response data flow to the driver interface process showing a successful update;
(h) if the map update process fails, send the update response data flow to the driver interface process showing an unsuccessful update;
(i) use the appropriate mechanism(s) such as a relational database, to write data to the store identified above;
(j) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.2;
USR = 1.3.2.1;

Output Flow Dynamics Assumptions:

driver_map_update_payment_request = driver_map_update_request;
driver_map_update_response = driver_map_update_request;
vehicle_map_database = driver_map_update_request;
tmup-vehicle_map_update_cost_request = driver_map_update_request;
tmup-vehicle_map_update_request = driver_map_update_request;

6.8.1.1.1 Determine Personal Portable Device Guidance Method

Input Flows

autonomous_traveler_guidance_data
dynamic_traveler_guidance_data
retained_traveler_guidance_data
traveler_guidance_accepted
traveler_guidance_data
traveler_guidance_request

Output Flows

autonomous_traveler_guidance_accepted
autonomous_traveler_guidance_data_request
dynamic_traveler_guidance_data_request
retained_traveler_guidance_data
traveler_guidance_instructions
traveler_input_request
traveler_route_accepted

Description:

Overview: This process shall act as the interface for personal guidance requests received from travelers with personal portable devices. The process shall select the best method for personal guidance based on data in the traveler's request Two methods shall be available to the process, comprising dynamic infrastructure based guidance is provided to the personal portable device), and autonomous (the personal portable device uses only locally available data- there is no information provided by the infrastructure) If the communications link to the central source fails, the process shall use the last set of guidance data that was received, and if this is not sufficient for the traveler to reach the requested destination, automatically revert to the use of autonomous guidance using local data only.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'traveler_guidance_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval from local data stores:
(a) 'autonomous_traveler_guidance_data';
(b) 'dynamic_traveler_guidance_data';
(c) 'retained_traveler_guidance_data';
(d) 'traveler_guidance_accepted';.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'autonomous_traveler_guidance_accepted';
(b) 'autonomous_traveler_guidance_data_request';
(c) 'dynamic_traveler_guidance_data_request';
(d) 'retained_traveler_guidance_data';
(e) 'traveler_input_request';
(f) 'traveler_guidance_instructions';
(g) 'traveler_route_accepted'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) select the appropriate process for the guidance data request dependent on whether the data input by the traveler specifies the use of dynamic guidance;
(c) if dynamic guidance is not available and was requested, then the autonomous guidance process must be used;
(d) if dynamic guidance is used but becomes unavailable then the autonomous guidance process must be used from the point at which the dynamic guidance was lost;
(e) if dynamic guidance having been lost is regained, the first dynamic guidance request must use the traveler's current location as the origin for the route request;
(f) the process shall be responsible for the maintenance of the store of data used in guidance requests using the appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.1;
USR = 1.3.1.2;

USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.2;
USR = 1.3.2.2;
USR = 1.3.2.3;
USR = 1.3.2.3.1;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.1;
USR = 1.3.4.1(d);
USR = 1.3.4.1(e);
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.2.2;
USR = 1.3.4.3;

Output Flow Dynamics Assumptions:

autonomous_traveler_guidance_accepted = AUTONOMOUS_TRAVS/(60);
autonomous_traveler_guidance_data_request = AUTONOMOUS_TRAVS/(60);
dynamic_traveler_guidance_data_request = DYNAMIC_TRAVS/(60);
retained_traveler_guidance_data = 2*ITS_TRAVS/(60*60);
traveler_guidance_instructions = ITS_TRAVS/(60);
traveler_input_request = ITS_TRAVS/(60);
traveler_route_accepted = DYNAMIC_TRAVS/(60);

6.8.1.1.2 Provide Personal Portable Device Dynamic Guidance

Input Flows

dynamic_traveler_guidance_data_request
traveler_guidance_route
traveler_location_for_dynamic_guidance

Output Flows

dynamic_traveler_guidance_data
traveler_route_request

Description:

Overview: This process shall enable dynamic traveler guidance data to be calculated. The process shall base its guidance request on the data input by the traveler from a personal portable device through other processes, and on the traveler's current location as provided by another process.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'dynamic_traveler_guidance_data_request';
(b) 'traveler_location_for_dynamic_guidance'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes:
(a) 'traveler_guidance_route'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'dynamic_traveler_guidance_data';
(b) 'traveler_route_request'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input data flow listed above;
(b) when the first input data flow in (a) is received send the traveler route request data flow;
(c) if no origin is specified in the guidance data request, then the current traveler location data obtained from the unsolicited input flow shall be used as the origin of the flow in (b);
(d) when the solicited input flow is received, load the data into the flow of dynamic traveler guidance data and sent it to the process that determines the traveler guidance method.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.2;
USR = 1.3.2.2;
USR = 1.3.2.3;
USR = 1.3.2.3.1;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.2.2;
USR = 1.3.4.3;

Output Flow Dynamics Assumptions:

traveler_route_request = 2/(60*60)*DYNAMIC_TRAVS;
dynamic_traveler_guidance_data = 2/(60*60)*DYNAMIC_TRAVS;

6.8.1.1.3 Provide Personal Portable Device Autonomous Guidance

Input Flows

autonomous_traveler_guidance_accepted
autonomous_traveler_guidance_data_request
traveler_location_for_autonomous_guidance
traveler_map_database

Output Flows

autonomous_traveler_guidance_data

Description:

Overview: This process shall provide autonomous on-line guidance when requested by the traveler from a personal portable device. It shall calculate the route using data obtained from a navigable map database stored in the traveler's personal portable device. Guidance shall be provided by the process in the form of actual instructions to the traveler, e.g. cross the road here, take the subway to a specific station. The process shall provide guidance for the shortest route, within the preferences and constraints specified by the traveler in the guidance request.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'autonomous_traveler_guidance_data_request';
(b) 'traveler_location_for_autonomous_guidance'.

Solicited Input Processing: This process shall receive the following data flows as a result of requests for data retrieval from local data stores and data being sent to other processes:
(a) 'autonomous_traveler_guidance_accepted';
(b) 'traveler_map_database'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'autonomous_traveler_guidance_data'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input data flows listed above;
(b) when the first input data flow in (a) is received, determine the shortest route using the data in the store of navigable map data;
(c) use the traveler location data in the second input data flow in (a) to determine the origin for the route unless one is specified in the traveler's request;
(d) when the route has been determined, read the associated guidance instructions from the store and output them to the process requesting guidance.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;
USR = 1.3.1;
USR = 1.3.1.1;
USR = 1.3.1.2;
USR = 1.3.1.2.1;
USR = 1.3.1.3;
USR = 1.3.2;
USR = 1.3.2.2;
USR = 1.3.2.3;
USR = 1.3.2.3.1;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.2;
USR = 1.3.3.3;
USR = 1.3.4;
USR = 1.3.4.2;
USR = 1.3.4.2.1;
USR = 1.3.4.2.2;
USR = 1.3.4.3;
USR = 1.5;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(d);
USR = 1.5.2.5(g);

Output Flow Dynamics Assumptions:

autonomous_traveler_guidance_data = 2/(60*60)*AUTONOMOUS_TRAVS;

6.8.1.2 Provide Personal Portable Device Guidance Interface

Input Flows

ft_guidance_data
ft_guidance_map_update_request
ft_guidance_request
ft_guidance_route_accepted
traveler_guidance_instructions
traveler_input_request
traveler_map_update_response
traveler_personal_credit_identity

Output Flows

traveler_guidance_accepted
traveler_guidance_data
traveler_guidance_request
traveler_map_update_request
traveler_personal_map_update_cost
tt_guidance
tt_guidance_input_request
tt_guidance_map_update_response
tt_guidance_route_details

Description:

Overview: This process shall be responsible for providing a user interface for the traveler through which personal guidance can be delivered. The process shall enable the traveler to input data to request a a suitable route. This process shall be capable of supporting two types of route guidance: dynamic (infrastructure based guidance is provided to the personal protable device), and autonomous (the personal portable device uses only locally available data- there is no information provided by the infrastructure). The process shall also act as the interface for output of on-line guidance to the traveler. Mutlimodal routes shall be supported by the process. The process shall not provide on-line guidance until the route has been accepted by the traveler. For those forms of guidance that require an on-board map database, the process shall provide an interface through which the traveler may obtain and pay for an initial copy of the database plus updates when needed. The process shall support inputs from the traveler in either manual or audio form, and shall provide outputs in audible or visual forms. It shall enable the visual output to be either in hardcopy, or display. Both types of output shall be produced in such a way that in using them the traveler does not become a hazard to other travelers.

Data Flows: All input data flows are unsolicited and all output flows are solicited

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows from the traveler listed above;
(b) when any inputs are received, generate the appropriate outputs identified above;
(c) implementation of guidance will generate a succession of output data which must be passed on to the traveler without the need for further input;
(d) the output in (c) must be retained until the next set of guidance data arrives for output;
(e) the advanced payment for map data flow is only generated when the traveler credit identity data flow contains a stored credit value and not a credit identity;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the credit identity or stored credit being transmitted using any form of digital or analogue technique.

User Service Requirements:

USR = 1.0;
USR = 1.3.0;

Output Flow Dynamics Assumptions:

traveler_guidance_accepted = ft-guidance_request;
traveler_guidance_data = 1/(60);
traveler_guidance_request = ft-guidance_request;
traveler_map_update_request = ft-guidance_map_update_request;
traveler_personal_map_update_cost = ft-guidance_map_update_request;
tt-guidance = 1/(60);
tt-guidance_input_request = 1/(60);
tt-guidance_map_update_response = ft-guidance_map_update_request;
tt-guidance_route_details = ft-guidance_request;

6.8.1.3 Process Personal Portable Device Location Data

Input Flows

From_Location_Data_Source
traveler_map_database

Output Flows

traveler_location_for_autonomous_guidance
traveler_location_for_dynamic_guidance
traveler_location_for_emergencies
traveler_location_for_information
traveler_location_for_planning

Description:

Overview: This process shall provide the traveler's current location. It shall calculate the location from one or more sources of position data such as GPS or DGPS, and shall refine its calculations using techniques such as map matching, dead reckoning, etc. The process shall provide the location to the to other processes for use in autonomous and dynamic guidance. This location should be precise as is practical within cost and technology constraints. It is intended for use by traveler personal navigation and guidance systems, as well as emergency notification systems.

Data Flows: The input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above;
(b) continuously compute the vehicle's current location using the data in the input flows and send it to other processes in the Provide Driver and Traveler Services and Manage Emergency Services functions using the output flows identified above;
(c) it shall be possible for the process to compute the location from as many sources of data as are simultaneously available to it, using the following priority order where more than one is available: differential GPS, GPS, map matching, magnetic flux and dead reckoning;
(d) the location determined by the source with the highest priority shall be used at all times, except that sources determined to be unreliable may be temporarily bypassed (e.g. GPS signals with low signal quality).

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

traveler_location_for_autonomous_guidance = 1;
traveler_location_for_emergencies = 1;
traveler_location_for_dynamic_guidance = 1;
traveler_location_for_information = 1;
traveler_location_for_planning = 1;

6.8.1.4 Update Traveler Navigable Map Database

Input Flows

fmup_traveler_map_update
fmup_traveler_map_update_cost
traveler_map_update_payment_response
traveler_map_update_request

Output Flows

tmup_traveler_map_update_cost_request
tmup_traveler_map_update_request
traveler_map_database
traveler_map_update_payment_request
traveler_map_update_response

Description:

Overview: This process shall update the traveler's navigable database based on digitized data obtained from a map provider, or other appropriate data source. The update shall be initiated by the traveler through another process. The process shall have the capability to allow a financial transaction (to pay for the update) to be completed using processes in the Provide Electronic Payment Services function. When the new map data is received, it shall be loaded by the process into the traveler_map_database data store for use by other processes. The result of the update request (successful or not) shall be sent back to the traveler interface process for output to the traveler.

Data Flows: The traveler update request input data flow is unsolicited. All other input flows and the output flows are solicited with the exception of the following which contains the new navigable map data to be written to its data store: 'traveler_map_database'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the traveler update request input flow listed above;
(b) when the input in (a) is received, send the data flow to the map update provider requesting the cost of the update;
(c) when a response to (b) is received and the credit identity not the stored credit was provided as part of the data in (a), generate the payment request;
(d) when a successful response is received to (c), generate the request to the map supplier for a new navigable map database;
(e) when a response to (b) is received and the stored credit not the credit identity was provided as part of the data in (a), compare it with the stored credit value and if greater send the update response data flow to the traveler interface process showing an unsuccessful update;
(f) if the check in (e) shows that there is sufficient stored credit, generate the request to the map supplier for a new navigable map database;
(g) when the response to the map database requests in (d) or (f) is received, load the new navigable map data into the data store, and send the update response data flow to the traveler interface process showing a successful update;
(h) if the map update process fails, send the update response data flow to the traveler interface process showing an unsuccessful update;
(i) use the appropriate mechanism(s) such as a relational database, to write data to the store identified above;

User Service Requirements:

USR = NA;

Output Flow Dynamics Assumptions:

traveler_map_database = traveler_map_update_request;(j) all input and output flows must be encrypted in such a way that it is not possible to determine
tmup-traveler_map_update_cost_request = traveler_map_update_request; the payment information being transmitted, using any form of digital or analog techniques.
tmup-traveler_map_update_request = traveler_map_update_request;
traveler_map_update_payment_request = traveler_map_update_request;
traveler_map_update_response = traveler_map_update_request;

6.8.1.5 Provide Traveler Emergency Message Interface

Input Flows

emergency_message_traveler_output
traveler_location_for_information

Output Flows

tt_emergency_message

Description:

Overview: This process shall provide an emergency notification interface for
a traveler using a personal portable device. The emergency notification interface shall enable the output of messages generated by a traveler's emergency request to another process.
Data Flows: All input data flows with the exception of that for traveler personal information and traveler location are unsolicited, but all output flows are solicited.
Functional Requirements: This process shall:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input in (a) is received, generate the traveler emergency message output flow identified above.

User Service Requirements:

USR = 1.0;
USR = 1.5.0;
USR = 1.5.2;
USR = 1.5.2.1;
USR = 1.5.2.2;
USR = 1.5.2.3;

Output Flow Dynamics Assumptions:

tt-emergency_message = emergency_message_traveler_output;

6.8.2.1 Build Traveler Personal Security Message

Input Flows

ft_personal_emergency_request
traveler_identity_store
traveler_location_for_emergencies

Output Flows

traveler_personal_emergency_request

Description:

Overview: This process shall respond to the input of a request from a traveler
for action by the emergency services. Input of the request shall be received by the process from the traveler via a panic button or some other functionally similar form of input device provided as part of the traveler's personal portable device. When the input is received, the process shall send a message to the communications process, containing the traveler's current location and identity.
(a) 'ft-emergency_request';
(b) 'traveler_location_for_emergencies'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'traveler_identity_store'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'traveler_personal_emergency_request'.

Functional Requirements: This process shall:
(a) continuously monitor arrival of input from the traveler;
(b) when input from the traveler is received obtain the current location and traveler identity and send this in a message to the communications process;
(c) if some or all of the data in (b) is missing, e.g. there is no current location, the message must be sent anyway, and repeated when a location becomes available.

User Service Requirements:

USR = 5.0;
USR = 5.1.0;
USR = 5.1.1;
USR = 5.1.1.1;
USR = 5.1.1.1(d);
USR = 5.1.1.1(e);
USR = 5.1.1.2;

Output Flow Dynamics Assumptions:

traveler_personal_emergency_request = ft-personal_emergency_request;

6.8.2.2 Provide Traveler Emergency Communications Function

Input Flows

emergency_request_personal_traveler_acknowledge
traveler_personal_emergency_request

Output Flows

emergency_message_traveler_output
emergency_request_personal_traveler_details

Description:

Overview: This process shall prepare and send an emergency message from a traveler's personal portable device to the Manage Emergency Services function. The message shall only be sent by the process in response to data received from another process that monitors traveler inputs. Once an emergency message has been sent, the process shall send a message to that effect to another process for output to the traveler. The process shall then await a response from the Manage Emergency Services function, and when received again send a message to the other process for output to the traveler. Output of the emergency message to the Manage Emergency Services function shall be repeated by the process at regular intervals until a response is received.
Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'traveler_personal_emergency_request'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to other processes and requests for data retrieval:
(a) 'emergency_request_personal_traveler_acknowledge'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'emergency_request_personal_traveler_details';
(b) 'emergency_message_traveler_output'.

Functional Requirements: This process shall:
(a) transmission of the output message must be as near to instantaneous as possible following the receipt of data from the security message process;
(b) the current data and time must be added to the data received from the security message process;
(c) when the acknowledgment message is received it should be sent immediately to the interface process for the personal traveler guidance facility;
(d) transmission of the output message must be repeated until an acknowledgment message is received updating the date and time as they change;
(e) initially, the message sent to the traveler must show that data has been sent, and only changed when an acknowledgment is received.

User Service Requirements:

USR = 5.0;
USR = 5.1.0;
USR = 5.1.1;
USR = 5.1.1.3;

Output Flow Dynamics Assumptions:

emergency_message_traveler_output = 1;
emergency_request_personal_traveler_details = 1/(60*60*24*7*52)*(ITS_GUIDED_TRAVS);

6.8.3.1 Get Traveler Personal Request

Input Flows

traveler_personal_trip_planning_requests

Output Flows

traveler_personal_current_condition_request
traveler_personal_payment_information
traveler_personal_traffic_condition_request
traveler_personal_transaction_request
traveler_personal_transit_condition_request
traveler_personal_trip_confirmation
traveler_personal_trip_request
traveler_personal_yellow_pages_information_request
traveler_traffic_profile
traveler_transit_profile

Description:

Overview: This process shall receive traveler requests from a personal device (portable, or non portable) then provide support for trip planning, traffic, transit and other (yellow pages) services information, trip confirmation, yellow pages services confirmation, and payment requests. The process shall send these requests to the appropriate processes within the Provide Driver and Traveler Services function for further processing to generate responses. The interface to the traveler shall be provided through a separate process, from which input to this process originates.

Data Flows: The input data flow is unsolicited and all output flows are solicited.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the traveler trip planning input flow listed above;
(b) when the flow in (a) is received, extract the data and send it to the appropriate processes in the Provide Driver and Traveler Services function;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.3;
USR = 1.1.3.2;
USR = 1.1.3.2.1;
USR = 1.1.3.2.2;
USR = 1.1.3.2.3;
USR = 1.1.3.2.4;
USR = 1.1.3.2.5;
USR = 1.1.3.2.6;
USR = 1.1.3.2.7;
USR = 1.1.3.2.8;
USR = 1.1.3.2.9;
USR = 1.1.3.2.10;
USR = 1.4.0;
USR = 1.4.1;
USR = 1.4.1.1;
USR = 1.4.1.2;
USR = 1.4.1.2(b);
USR = 1.4.1.2(c);
USR = 1.4.1.3;
USR = 1.5.0;
USR = 1.5.2;

Output Flow Dynamics Assumptions:

traveler_personal_current_condition_request = traveler_personal_trip_planning_requests;
traveler_personal_payment_information = (traveler_personal_trip_planning_requests)/5;
traveler_personal_traffic_condition_request = traveler_personal_trip_planning_requests;
traveler_personal_transaction_request = (traveler_personal_trip_planning_requests)/5;
traveler_personal_transit_condition_request = traveler_personal_trip_planning_requests;
traveler_personal_trip_confirmation = (traveler_personal_trip_planning_requests)/5;
traveler_personal_trip_request = traveler_personal_trip_planning_requests;

traveler_personal_yellow_pages_information_request = traveler_personal_trip_planning_requests;
traveler_transit_profile = 1/(60*60*24*7);
traveler_traffic_profile = 1/(60*60*24*7);

6.8.3.2 Provide Traveler with Personal Travel Information

Input Flows

map_data_for_traveler_personal_displays
traffic_data_for_broadcast_to_personal_devices
traffic_data_for_personal_devices
transit_deviations_for_broadcast_to_personal_devices
transit_deviations_for_personal_devices
transit_services_for_personal_devices
traveler_personal_payment_confirmation
traveler_personal_traffic_condition_request
traveler_personal_transaction_confirmation
traveler_personal_transit_condition_request
traveler_personal_trip_information
traveler_personal_yellow_pages_data

Output Flows

traffic_data_personal_request
transit_deviations_personal_request
transit_services_personal_request
traveler_personal_trip_planning_responses

Description:

Overview: This process shall provide the traveler (using a personal device) with data about all requested trip, traffic, transit, other (yellow pages) services information, confirmation of any requested reservations, and payments made as part of confirmed trip plans. The data shall be sent by the process to an interface process which is responsible for its actual output to the traveler. This data shall include digitized map data to act as the background to the output when the data is shown in a suitable format. This process shall request data from other ITS functions or be sent it as a result of requests from another process.

exception of the following:
(a) 'traffic_data_for_portables' - which is received as a result of output being sent to another process;
(b) 'transit_deviations_for_portables' - which is received as a result of output being sent to another process;
(c) 'transit_services_portables_request' - which is received as a result of output being sent to another process;
(d) 'map_data_for_traveler_personal_displays' - which contains data requested from a data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above that are not details of transit services, traffic data and the display map data;
(b) when any of the flows in (a) are received, retrieve the relevant digitized display map data from the local store and send the combined data to the traveler interface process;
(c) when the flow received in (a) contains a request for transit or traffic data, send the request to the relevant process in the Manage Transit or Manage Traffic function;
(d) the input data received as a result of (c) shall be combined with the relevant digitized display map data from the local store and sent to the traveler interface process;
(e) use the most appropriate mechanism(s) such as a relational database, to retrieve data from the store identified above;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.1;
USR = 1.1.1.1;
USR = 1.1.1.1.1;
USR = 1.1.1.1.2;
USR = 1.1.1.1.3;
USR = 1.1.1.1.4;
USR = 1.1.1.1.5;
USR = 1.1.1.1.6;
USR = 1.1.2;
USR = 1.1.2.1;

Output Flow Dynamics Assumptions:

transit_deviations_personal_request = traveler_personal_trip_planning_requests;
traffic_data_personal_request = traveler_personal_trip_planning_requests;
transit_services_personal_request = traveler_personal_trip_planning_requests;
traveler_personal_trip_planning_responses = traveler_personal_trip_planning_requests;

6.8.3.3 Provide Traveler Personal Interface

Input Flows

ft_personal_extra_trip_data
ft_personal_map_display_update_request
ft_personal_trip_planning_requests
traveler_location_for_planning
traveler_personal_credit_identity
traveler_personal_display_map_update_response
traveler_personal_regular_data
traveler_personal_trip_planning_responses

Output Flows

traveler_personal_display_map_update_request
traveler_personal_display_update_cost
traveler_personal_regular_data
traveler_personal_trip_costs
traveler_personal_trip_planning_requests
tt_personal_extra_trip_data_request
tt_personal_trip_planning_responses

Description:

Overview: This process shall provide an interface in a personal device through which travelers can plan and confirm trips, as well as obtain current traffic and transit information. The process shall support trip planning and confirmation of other (yellow pages) services such as lodging, restaurants, theaters, and other tourist activities. The process shall be able to load in the traveler_personal_regular_data store frequently used information such as traveler identity (the owner of the personal device), home and work locations, etc. This will reduce the amount of input needed by the traveler for each trip request. The process shall also carry out input data verification and require input confirmation, with the traveler, before passing the data to other processes. The traveler's payment information and location (when traveler is using a portable device) shall be obtained by this process from other processes. The process shall support inputs from the traveler in both manual and audio form, and shall provide its outputs in audible and visual forms that are consistent with a personal device. This process shall include forms suitable for travelers with hearing and vision physical disabilities. The process shall display data for as long as required by the traveler and must enable viewing of previously output data. When used with a portable device, the process shall provide the traveler the option to filter the data (to be displayed) relevant to the travelers current location.

Data Flows: All input data flows are unsolicited and all output flows are solicited, with the exception of the 'traveler_personal_regular_data' data flow which contains data requested from or written to a data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows from the traveler listed above;
(b) when any of the inputs in (a) are received, check for content, accuracy, consistency and out of range values, utilizing data from the local store identified above if necessary;
(c) generate the output identified above and load the requested data into the local data store;
(d) continually monitor the data in the local store and compare it with that being input by travelers, deleting any data from the store which is not frequently used;
(e) be responsible for the management of the data in the store of regularly used data, using the appropriate mechanism(s) such as a relational database, for storing the data;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.1.0;
USR = 1.1.3;
USR = 1.1.3.2;
USR = 1.1.3.2.1;
USR = 1.1.3.2.2;
USR = 1.1.3.2.3;
USR = 1.1.3.2.4;
USR = 1.1.3.2.5;
USR = 1.1.3.2.6;
USR = 1.1.3.2.7;
USR = 1.1.3.2.8;
USR = 1.1.3.2.9;

Output Flow Dynamics Assumptions:

traveler_personal_display_map_update_request = ft-personal_map_display_update_request;
traveler_personal_display_update_cost = ft-personal_map_display_update_request;
traveler_personal_regular_data = (ft-personal_trip_planning_requests)/10;
traveler_personal_trip_costs = ft-personal_trip_planning_requests;
traveler_personal_trip_planning_requests = ft-personal_trip_planning_requests;
tt-personal_extra_trip_data_request = ft-personal_extra_trip_data;
tt-personal_trip_planning_responses = ft-personal_trip_planning_requests;

6.8.3.4 Update Traveler Personal Display Map Data

Input Flows

fmup_traveler_personal_display_update
fmup_traveler_personal_display_update_cost
traveler_personal_display_map_update_request
traveler_personal_display_update_payment_response
traveler_personal_regular_data

Output Flows

map_data_for_traveler_personal_displays
tmup_request_traveler_personal_display_update
tmup_request_traveler_personal_display_update_cost
traveler_personal_display_map_update_response
traveler_personal_display_update_payment_request

Description:

Overview: This process shall provide updates to the digitized map data used as the background for displays on travelers' personal devices. These displays include details of traffic, trip and travel information for use by travelers. The process shall obtain the new map data from a map provider process or some other appropriate data source on request from the traveler via the traveler interface process. The process shall load data into the map_data_for_traveler_personal_displays

data store. The data will be compatible with the types of displays that are found on personal devices.

Unsolicited Input Processing: This process shall receive the following unsolicited input data flows:
(a) 'request_traveler_personal_display_map_update'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to an external function (terminator):
(a) 'fmup-traveler_display_update';
(b) 'fmup-traveler_personal_display_update_cost'.

Solicited Input Processing: This process shall receive the following data flows as a result of output being sent to another process:
(a) 'traveler_personal_display_update_payment_response'.

Solicited Input Processing: This process shall receive the following data flows as a result of a request for data from a store:
(a) 'traveler_personal_regular_data'.

Solicited Output Processing: This process shall provide the following output flows as a result of the above inputs being received:
(a) 'map_data_for_traveler_personal_displays';
(b) 'tmup-request_traveler_personal_display_update';
(c) 'tmup-request_traveler_personal_display_update_cost';
(d) 'traveler_personal_display_update_payment_request';
(e) 'traveler_personal_display_map_update_response'.

Functional Requirements: This process shall:
(a) monitor for the receipt of the unsolicited input data flow shown above;
(b) when the flow in (a) is received, send out the request for new data from the specialized digital map data supplier;
(c) be capable of receiving the input data in a variety of formats and converting it into a single format suitable for use with the store of digitized map data;
(d) use the appropriate mechanism(s), such as a relational database, to retrieve data from the store identified above;
(e) be responsible for the management of the data in the store of digitized map data, using the appropriate mechanism(s), such as a relational database, for storing the data.

User Service Requirements:

USR = 1.3;
USR = 1.3.4;
USR = 1.3.4.1;
USR = 1.3.4.1(b);
USR = 1.3.4.1(c);

USR = 1.5;
USR = 1.5.2;
USR = 1.5.2.5;
USR = 1.5.2.5(c);
USR = 1.5.2.5(d);

Output Flow Dynamics Assumptions:

map_data_for_traveler_personal_displays = traveler_personal_display_map_update_request;
tmup-request_traveler_personal_display_update = traveler_personal_display_map_update_request;
tmup-request_traveler_personal_display_update_cost = traveler_personal_display_map_update_request;
traveler_personal_display_update_payment_request = traveler_personal_display_map_update_request;
traveler_personal_display_map_update_response = traveler_personal_display_map_update_request;

7.1.1.1 Read Tag Data for Tolls

Input Flows

toll_tag_data_collect
vehicle_toll_characteristic_data

Output Flows

get_toll_tag_violator_image
toll_tag_data_request
toll_tag_data_update
toll_tag_problem_message
vehicle_tag_for_tolls
vehicle_type_for_tolls

Description:

Overview: This process shall be responsible for requesting the data from the toll tag being carried on-board the vehicle and used as a payment instrument. If there is no tag or the data it contains cannot be properly read, this process shall provide a message for the vehicle operator to contact the toll authority (or toll system operator). The process shall send a request to other processes to obtain an image of the vehicle. If the vehicle is exiting a closed toll system the data shall be checked by this process to see if it contains an entry point toll segment number. If not present, the process would be referred to another process for off-line resolution. If the toll segment identity is present, it shall be combined with the vehicle characteristics, e.g., size, type, etc., to form the data upon which the toll payment transaction can be based, and the data sent to another process. If the vehicle is entering a closed toll system, the entry point toll segment shall be written onto the tag so that it can be used as the mechanism for charging for the use of the toll road.

Data flows: All input and output data flows are solicited with the exception of the following item which is used to trigger the process:
(a) 'vehicle_toll_characteristic_data' - which is received from another process that detects a vehicle's presence.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the unsolicited input identified above is received, generate the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.13;
USR = 1.8.2.13(c);
USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.1;
USR = 3.1.1.6;

Output Flow Dynamics Assumptions:

get_toll_tag_violator_image = TAG_DEFECT_RATE*(vehicle_toll_characteristic_data);
toll_tag_data_request = vehicle_toll_characteristic_data;
toll_tag_data_update = vehicle_toll_characteristic_data;
toll_tag_problem_message = TAG_DEFECT_RATE*(vehicle_toll_characteristic_data);
vehicle_tag_for_tolls = vehicle_toll_characteristic_data;
vehicle_type_for_tolls = vehicle_toll_characteristic_data;

7.1.1.10 Determine Advanced Toll Bill

Input Flows

advanced_toll_needed
toll_price_data_for_advanced_toll

Output Flows

advanced_toll_billing

Description:

Overview: This process shall be responsible for receiving a request to pay an advanced toll. It shall obtain the price of the toll segment(s) for which advanced payment is being requested from a local data store and shall then forward it to the billing processes. The store of toll prices shall be maintained by another process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(d) use the most appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.1;

Output Flow Dynamics Assumptions:

advanced_toll_billing =
15/60*TOLL_LANES+4/60*PARKING_LANES*PARKING_LOTS+10/60*ITS_TRANSIT_VEHS;

7.1.1.11 Manage Toll Archive Data

Input Flows

fta_archive_commands
toll_archive_request
toll_archive_status
toll_data_archive
toll_operational_data
toll_prices_for_archive

Output Flows

toll_archive_data
toll_data_archive
tta_archive_status

Description:

Overview: This process shall obtain toll operational data and toll pricing data and distribute it to the Manage Archived Data function. As data is received into this process quality control metrics shall be assigned. The appropriate meta-data shall be generated and store along with the data. A catalog of the data shall be maintained to allow requesters to know what data is available from the archive store. The process shall run when a request for data is received from an external source, or when fresh data is received.

All inputs to this process are unsolicited, and all outputs are solicited, except that the 'toll_archive_status' is a solicited input.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when any of the unsolicited data inputs shown above is received, the process shall store them in the data store along with meta data (data attributes about the data), and update the catalog;
(c) when the unsolicited input from the toll administrator is received, the process shall update the data store accordingly;
(d) when the request for toll archive data is received, the process shall immediately generate the solicited output shown above from the data store;
(e) the process should then receive the toll archive status solicited input and send this status to the toll adminstrator;
(f) data shall only be sent to the source from which the data request originated;
(g) before output, the process shall put the data into a format that is easily read and interpreted by external processes and can also be read by travelers and toll users with the minimum of further processing.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.3;
USR = 7.1.3.1.2;

Output Flow Dynamics Assumptions:

toll_archive_data = toll_archive_request;
tta-archive_status = 1/(60*60*24);

7.1.1.2 Calculate Vehicle Toll

Input Flows

fto_local_toll_price_variations
toll_price_data_for_vehicle_toll
vehicle_tag_for_tolls
vehicle_type_for_tolls

Output Flows

toll_charge

Description:

Overview: This process shall be responsible for calculating the toll for the detected vehicle based on the vehicle's characteristics and data obtained from the tag being carried by the vehicle. This process shall calculate the cost of the toll using segment(s) traveled by the vehicle. Segment information is obtained by reading data that contains standard prices for toll segments plus any variations to pricing received from the toll operator.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) use the most appropriate mechanism(s) such as a relational database, to retrieve data from the store identified above;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.13;
USR = 1.8.2.13(c);
USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.1;
USR = 3.1.1.2;
USR = 3.1.1.6;

Output Flow Dynamics Assumptions:

toll_charge = 30/(60)*TOLL_LANES;

7.1.1.3 Manage Bad Toll Payment Data

Input Flows

bad_toll_payment_list
ffi_bad_toll_payment_updates
toll_bad_payment_check_request
toll_payment_violator_data

Output Flows

bad_toll_payment_list
tfi_toll_payment_violator_data
toll_bad_payment_check_response

Description:

Overview: This process shall be responsible for maintaining a data store containing a list of invalid driver credit identities. This process shall use this data to verify credit identities and commercial vehicle carrier numbers provided for checking by the billing process. Verification shall ensure that the current toll payment transaction is using a credit identity or carrier identity that has not previously had an invalid transaction. Details of potential invalid credit identities or carrier numbers shall be sent by this process to the financial institution for verification. This process shall also receive from the financial institution details of invalid payment instrument data that has been found by other means.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data written to or requested from a data store:
(a) 'bad_toll_payment_list'.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) be responsible for the management of the data in the store of bad driver bad_toll_payment_list data, using the appropriate mechanism(s) such as a relational database, for storing the data.
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.1;
USR = 1.8.2.1(f);
USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;
USR = 3.1.1.5;

Output Flow Dynamics Assumptions:

tfi-toll_payment_violator_data = 8/(60*60*24)*TOLL_LANES;
toll_bad_payment_check_response = toll_bad_payment_check_request;
bad_toll_payment_list = 8/(60*60*24)*TOLL_LANES;

7.1.1.4 Check for Advanced Tolls Payment

Input Flows

advanced_toll_billing
advanced_toll_payment_list
toll_charge

Output Flows

advanced_toll_payment_list
advanced_toll_transactions
billing_for_tolls_needed
tto_transaction_reports

Description:

Overview: This process shall be responsible for checking to see if the required toll payment has already been made. The process shall determine the existence of an advanced payment for the toll segment(s) by comparing the received payment information with that in the store containing the list of advanced payments. If the payment has already been made then the process shall remove the requirement for local billing and remove the record of the advanced payment from the store. Details of each payment transaction shall be sent by the process to another process with the payment information received from the driver removed.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'advanced_toll_payment_list'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) be responsible for the management of the data in the store of advanced tolls payment data, using the most appropriate mechanism(s) such as RDBMS, for storing the data;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(e) remove the payment information received from the driver from all data that is sent to another process for loading into the store of toll payment transactions.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.5;

Output Flow Dynamics Assumptions:

advanced_toll_transactions = 25/(60)*TOLL_LANES;
billing_for_tolls_needed = 20/(60)*TOLL_LANES;
tto-transaction_reports = 1/(60*60)*TOLL_LANES;

7.1.1.5 Bill Driver for Tolls

Input Flows

billing_for_tolls_needed
toll_bad_payment_check_response
toll_payment_confirmation

Output Flows

advanced_toll_payment_list
confirm_advanced_tolls_payment
current_toll_transactions
get_toll_payment_violator_image
toll_bad_payment_check_request
toll_payment_debited
toll_payment_pull_in_message
toll_payment_request
toll_payment_violator_data
toll_tag_data_clear

Description:

Overview: This process shall be responsible for either obtaining payment for the current or advanced toll. The process shall achieve this either by requesting that the toll cost be deducted from the credit being stored by the toll tag that is acting as the payment instrument, or by informing the driver that payment for the toll will be debited to the credit identity provided by the tag. Before sending data to the tag, the process shall check that either the credit identity is not already in the list of bad payers, or the stored credit is not less that the toll cost. If either of these conditions is true, the process shall obtain an image of the driver and vehicle which can be forwarded to the appropriate enforcement agency via another process. When the appropriate payment transaction has been completed, the toll entry segment identity shall be cleared from the tag so that it can be used the next time that the vehicle is on a toll road. The tag may be in the form of some type of credit or debit card, or an electronic purse. Details of the transaction shall always be sent by this process to the process that manages toll transactions. Where an advanced toll payment is identified, the process shall take no action if the credit identity is on the bad payers list, or the stored credit is less than the toll cost, other than the payment is not confirmed.

exception of the following:
(a) 'toll_bad_payment_check_response', which contains data from another process;
(b) 'advanced_toll_payment_list', which contains data written to a data store.

Functional Requirements: This process shall:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) when the unsolicited flow requesting billing for tolls is received, check to see if it contains a credit identity, carrier identity or a stored credit value;
(d) if a credit identity or carrier identity is found in (c) send it to the process managing the store of bad payees for a check that it is not on the list of bad toll payments;
(e) if a match is found in (d), get an image of the violator and output the data flow that requests the vehicle to pull in;
(f) if a stored credit value is found in (c) and it is less that the toll cost, get an image of the violator and output the data flow that requests the vehicle to pull in;
(g) if the stored credit value found in (c) is greater than or equal to the toll cost, send the output flow to the payment instrument requesting that the toll cost be deducted from the credit being stored by the instrument;
(h) if a negative response is received to (g), get an image of the violator and output the data flow that requests the vehicle to pull in;
(i) when the toll transaction is complete always output details of the transaction in the flow of current toll transactions and send out the data flow that clears the toll tag data store;
(j) if the payment is identified as being for an advanced toll, and a match if found in (d) or the stored credit value is less than the toll cost then set the output flow of advanced toll payment to false and take no further action;
(k) if the tests if (j) are clear then set the output flow of advanced toll payment to true and enter the credit and vehicle identities into the store containing the advanced payment list;
(l) use the appropriate mechanism(s) such as a relational database, to retrieve and write data to the store identified above;
(m) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.1;
USR = 3.1.1.8;

Output Flow Dynamics Assumptions:

advanced_toll_payment_list =
15/60*TOLL_LANES+4/60*PARKING_LANES*PARKING_LOTS+1/(60*60)*ITS_TRAVS;
toll_bad_payment_check_request = 20/(60)*TOLL_LANES;
confirm_advanced_tolls_payment = 15/(60)*TOLL_LANES;
current_toll_transactions = 20/(60)*TOLL_LANES;
get_toll_payment_violator_image = 8/(60*60*24)*TOLL_LANES;
toll_payment_request = 20/(60)*TOLL_LANES;
toll_payment_debited = 20/(60)*TOLL_LANES;
toll_payment_pull_in_message = 20/(60)*TOLL_LANES;
toll_payment_violator_data = 1/(60*60)*TOLL_LANES;
toll_tag_data_clear = 20/(60)*TOLL_LANES;

7.1.1.6 Collect Probe Data From Toll Transactions

Input Flows

toll_transactions_for_probe_data

Output Flows

probe_data_for_traffic
toll_transactions_for_probe_data_request
vehicle_toll_probe_data

Description:

Overview: This process shall calculate the time taken for vehicles to travel
between successive toll plazas and send it to the Manage Traffic and Provide Driver and Traveler Services functions. The process shall periodically request the data from the process that manages toll financial processing and ensure that any references to the driver and/or vehicle identity plus any other payment information are removed from the data before it is sent to the other functions.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall:
(a) periodically generate the toll_transactions_for_probe_data request data flow shown above;
(b) when the toll_transactions_for_probe_data flow is received, calculate the average journey times between toll plazas where there is sufficient data to enable an average to be sensibly computed;
(c) when (b) is completed, generate the two outputs of probe data identified above;
(d) remove any credit and/or vehicle identity plus other payment information from the data obtained in the input data flow identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.1;
USR = 3.1.1.4;
USR = 3.1.1.6;

Output Flow Dynamics Assumptions:

probe_data_for_traffic = 12/(60*60);
toll_transactions_for_probe_data_request = 12/(60*60);
vehicle_toll_probe_data = 12/(60*60);

7.1.1.7 Update Toll Price Data

Input Flows

fta_toll_price_changes_response
fta_toll_price_data
toll_price_changes_request
toll_price_data_request
toll_price_direct_request
toll_prices

Output Flows

toll_price_changes_response
toll_price_data
toll_price_data_for_advanced_toll
toll_price_data_for_vehicle_toll
toll_price_direct_details
toll_prices
toll_prices_for_archive
tta_toll_price_changes_request

Description:

Overview: This process shall be responsible for maintaining a store of data containing the toll price, which may vary according to the type of vehicle. The process shall also act as the interface for the output and input of responses to toll price change requests from the Manage Traffic function, the provision of toll price information to the Centralized Payments facility, and to the toll administrator to enable changes to be made to the stored data. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'toll_prices'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when any of the inputs from the toll administrator are received, generate the appropriate output identified above;
(c) when the input for a toll prices change request is received from the Manage Transit function, generate the change request output to the toll service provider;
(d) when a request for toll price data is received, generate the data flow containing the copy of the store of price data;
(e) be responsible for the management of the data in the store of toll cost data, using the most appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.2;

Output Flow Dynamics Assumptions:

toll_price_changes_response = 4/(60*60);
toll_prices = 12/(60*60*24*7*52)*TOLL_PLAZAS;
toll_price_data = toll_price_data_request+ftsp-toll_price_data;

7.1.1.8 Register for Advanced Toll Payment

Input Flows

advanced_other_tolls_request
advanced_traveler_tolls_request
confirm_advanced_tolls_payment
fta_confirm_advanced_toll

Output Flows

advanced_other_tolls_confirm
advanced_toll_needed
advanced_traveler_tolls_confirm
tta_request_advanced_toll

Description:

Overview: This process shall be responsible for responding to requests for tolls to be paid in advance. It shall provide the toll administrator with the opportunity to review the requests for advanced toll payments. If approved, the advanced toll data shall be forwarded by the process to other processes for the actual toll cost to be obtained and payment transactions initiated.
Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs requesting advanced tolls are received, generate the outputs to the toll administrator identified above;
(c) if no response is received to the flows output in (b), assume that the advanced tolls have been accepted and send the data to the advanced toll bill determination process;
(d) if a negative response is received to the flows in (b), then output the advanced toll response data flows with the confirmation data set to fail;
(e) when the confirm data flow is received from the bill driver for tolls process, then output the advanced toll response data flows with the confirmation data set to true;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.1;

Output Flow Dynamics Assumptions:

advanced_other_tolls_confirm = 4/60*PARKING_LANES*PARKING_LOTS+1/(60*60)*ITS_TRAVS;
advanced_toll_needed =
15/60*TOLL_LANES+4/60*PARKING_LANES*PARKING_LOTS+1/(60*60)*ITS_TRAVS;
advanced_traveler_tolls_confirm = 1/(60*60)*ITS_TRAVS;
tta-request_advanced_toll =
tta-toll_price_changes_request = 4/(60*60);
toll_prices_for_archive = 4/(60*60);
toll_price_direct_details = toll_price_direct_request;
toll_price_data_for_advanced_toll = 4/(60*60);
toll_price_data_for_vehicle_toll = 4/(60*60);

7.1.1.9 Manage Toll Financial Processing

Input Flows

advanced_toll_transactions
current_toll_transactions
ffi_confirm_toll_payment
toll_transaction_records
toll_transactions_for_probe_data_request

Output Flows

tfi_request_toll_payment
toll_operational_data
toll_transaction_records
toll_transactions_for_probe_data
tta_transaction_reports

Description:

Overview: This process shall be responsible for maintaining a log of all toll transactions that are carried out by other processes in the toll payments system. At periodic intervals the process shall output the accumulated records to the toll administrator and the archive function. It shall also output the data on request to the process that calculates probe data from the average travel time between toll plazas. The identity of the payee shall be removed from the data before it is used in any of these outputs. The process shall also be responsible for sending details of transactions to the financial institution to enable the users to be billed through their credit identities. For commercial vehicles, this will be done using the data provided by the vehicle's on-board tag and shall enable billing to the financial institution to be made by carrier.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'toll_operational_data' - which is an unsolicited data flow periodically sent to the Manage Toll Archive Data function.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs of transaction records are received, load the data in the data store served by the 'inout' flow shown in the above list;
(c) when requested by the appropriate input, generate the outputs identified above;
(d) be responsible for the management of the data in the store of toll transaction records, using the most appropriate mechanism(s) such as RDBMS, for storing the data;
(e) remove all driver identities from the data before it is used in any output data flows;
(f) ensure that all output flows are encrypted in such a way that it is not possible to determine the financial data being transmitted, using any form of digital or analog techniques;
(g) periodically (not less than once per day) send the toll operational data flow to the archive function calculating the number of users for each toll segment from the collected toll costs.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.4;
USR = 1.8.2.4(e);
USR = 1.8.2.10;
USR = 1.8.2.10(a);
USR = 1.8.2.12;
USR = 1.8.2.12(a);
USR = 3.0;
USR = 3.1;

Output Flow Dynamics Assumptions:

tfi-request_toll_payment = 1/(60*60*24)*TOLL_PLAZAS;
toll_operational_data = 1/(60*60*24);
toll_transactions_for_probe_data = toll_transactions_for_probe_data_request;
toll_transaction_records = advanced_toll_transactions + current_toll_transactions;
tta-transaction_reports = 1/(60*60)*TOLL_PLAZAS;

7.1.2 Produce Roadside Displays

Input Flows

toll_payment_pull_in_message
toll_tag_problem_message

Output Flows

td_toll_payment_confirmed
td_toll_payment_invalid

Description:

Overview: This process shall be responsible for driving the displays that tell vehicles whether or not their driver's toll payment has been confirmed or rejected. The process shall receive the data for output via the displays from other processes. The data input and output forms shall use an appropriate form of display that shall be easily readable under all lighting conditions and over the range of speeds that vehicles are expected to use when passing through the toll plaza. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

; Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when either of the inputs is received, generate the appropriate output identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.3;
USR = 3.1.4
USR = 3.1.4.3;

Output Flow Dynamics Assumptions:

td-toll_payment_confirmed = 80/(60)*TOLL_LANES;
td-toll_payment_invalid = (1/(60*60)+16/(60*60*24))*TOLL_LANES;

7.1.3 Obtain Toll Violator Image

Input Flows

From_Vehicle_Characteristics
get_toll_payment_violator_image
get_toll_tag_violator_image

Output Flows

toll_violation_information

Description:

Overview: This process shall be responsible for obtaining an image of a violator for use by other processes. The form of the image data obtained by this process shall be very accurate so that there can be no mistake of the determination of the identity of the vehicle and/or driver, and shall be easily passed on by the other processes to the appropriate law enforcement agency(ies) so that punitive action may be taken. The process shall be capable of obtaining an image of the required accuracy under all lighting conditions and over the range of speeds with which vehicles will pass through the toll plaza.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when the input data flows requesting a violator image be obtained are received, read the vehicle characteristics data being received;
(c) use the data in (b) to generate a highly accurate image of the vehicle and/or its driver;
(d) output the image in the violation information flow identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;

Output Flow Dynamics Assumptions:

toll_violation_information = (1/(60*60)+16/(60*60*24))*TOLL_LANES;

7.1.4 Provide Driver Toll Payment Interface

Input Flows

advanced_fares_and_charges_response
driver_toll_payment_credit_identity
fbv_vehicle_identity
fd_other_services_toll_request

Output Flows

advanced_fares_and_charges_request
driver_advanced_payment_at_toll
td_other_services_toll_response

Description:

Overview: This process shall be responsible for providing an interface through which drivers can request and pay for other services when paying their tolls at toll plazas. The services supported by this process include advanced payment for parking lot charges and transit fares. The process shall query the driver for sufficient information to enable the advanced parking lot charge and/or transit fare to be determined and the cost either billed to a credit identity provided by the driver's payment instrument, or deducted from credit stored on the instrument. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the driver, the basic vehicle and the driver's payment credit identity input flows are received, check the content of the input data to ensure that there is sufficient data to enable the advanced parking lot charge and/or transit fare to be determined;
(c) if the check in (b) is not confirmed, output an insufficient data to complete transaction message to the driver using the appropriate output flow and highlighting the missing data;
(d) the data required in (b) about the advanced booking shall include such things as the identity of the parking lot plus the date and time for which a space is required, and/or the origin and destination of the transit route;
(e) the data in (b) must in addition to that for the advanced booking request, also include the vehicle identity and either the driver's credit identity or the credit, stored on the payment instrument being used by the driver;
(f) if the check in (f) is not confirmed, output an insufficient financial data (credit identity or stored credit) to complete transaction message to the driver using the appropriate output flow;
(g) if advanced parking lot booking is required output the appropriate flow to generate any required charges only, using the new value of the stored credit if advanced tolls were required;
(h) if advanced parking lot booking was required in (g), when a response is received again reduce the value of the stored credit;
(i) if an advanced transit fare is to be paid, output the appropriate flow to generate any advanced transit fares, using the new value of the stored credit generated by any advanced costs already produced from the above;
(j) when output and response to all the advanced booking requests are complete, total up all the costs and check that they are less than the original value of the stored credit and if not output an insufficient credit message to the driver using the appropriate output flow;
(k) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;

Output Flow Dynamics Assumptions:

advanced_fares_and_charges_request = fd-other_services_toll_request;
driver_advanced_payment_at_toll = fd-other_services_toll_request;
td-other_services_toll_response = fd-other_services_toll_request;

7.1.5 Detect Vehicle for Tolls

Input Flows

From_Vehicle_Characteristics

Output Flows

vehicle_toll_characteristic_data

Description:

Overview: This process shall be responsible for producing a vehicle's characteristics from data received by sensors located at the roadside, at or near the toll collection point. The data shall be sent by the process to another process in a form suitable for use in calculating the toll cost for the vehicle. The process shall ensure that the data includes such things as vehicle size, weight, axle count, type, identifiable features, etc.

Data Flows: Input data flow is unsolicited and output flow is solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flow listed above;
(b) when the input is received, generate the output data flow identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.4;
USR = 3.1.1.8;

Output Flow Dynamics Assumptions:

vehicle_toll_characteristic_data = 30/(60)*TOLL_LANES;
USR = 3.1.1.1;
USR = 3.1.1.3;
USR = 3.1.1.7;

7.1.6 Distribute Advanced Charges and Fares

Input Flows

advanced_fares_and_charges_request
advanced_other_tolls_confirm
transfer_charges_to_tolls
transfer_fares_to_tolls

Output Flows

advanced_fares_and_charges_response
advanced_other_tolls_request
transfer_tolls_to_charges
transfer_tolls_to_fares

Description:

Overview: This process shall be responsible for receiving requests for advanced payment of tolls from the parking lot charge or transit fare collection facilities within the Provide Electronic Payment Services function. It shall pass the requests on to another process in the toll collection facility, and shall return transaction success or failure details to the requesting process. The process shall also receive requests for the advanced payment of parking lot charges and transit fares from the toll payment interface process. It shall send these requests to other processes in the Provide Electronic Payment Services function and when received, return the results to the toll payment interface process.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs from other processes in the Provide Electronic Payment Services function are received, generate the advanced toll payment request output identified above;
(c) when a response is received to the flow in (b), return it in the flow to the requesting process in the Provide Electronic Payment Services function;
(d) when the inputs requesting advanced parking lot charge or transit fare payment are received from the toll payment interface process, generate the appropriate transfer flows to other processes in the Provide Electronic Payment Services function;
(e) when a response is received to the flows in (d), return it in the flow to the requesting process;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

advanced_fares_and_charges_response = 10/(60)*TOLL_LANES*TOLL_PLAZAS;
advanced_other_tolls_request =
4/(60*60)*PARKING_LANES*PARKING_LOTS+10/(60*60)*ITS_TRANSIT_VEHS;
transfer_tolls_to_charges = 10/(60)*TOLL_LANES*TOLL_PLAZAS;

7.1.7 Provide Payment Instrument Interface for Tolls

Input Flows

driver_advanced_payment_at_toll
fpi_confirm_payment_at_toll_plaza
fpi_toll_tag_data
toll_payment_debited
toll_payment_request
toll_tag_data_clear
toll_tag_data_needed
toll_tag_data_request
toll_tag_data_store
toll_tag_data_update

Output Flows

driver_toll_payment_credit_identity
toll_payment_confirmation
toll_tag_data_collect
toll_tag_data_input
toll_tag_data_store
tpi_debited_payment_at_toll_plaza
tpi_request_payment_at_toll_plaza

Description:

Overview: This process shall be responsible for providing the interface through which the payment information can be read from a vehicle tag. The process shall enable the use of the data from the tag for the purposes of paying for current tolls, plus if required, the cost of advanced parking lot charges, and/or transit fares, as well as providing the data for use in traffic flow analysis. The tag data which can be collected by the process shall include credit identity, stored credit value, and the toll segment identity at the vehicle's entry point so that closed toll system can be used. When stored credit is used, the process shall enable the deduction of the cost of the toll and (possibly) advanced payments from the credit value on the tag. The process shall support collection of data from tags on-board a range of vehicle types including private cars or vans, commercial vehicles, transit vehicles, including those used for demand responsive transit services.

Data Flows: All input and output data flows are solicited with the exception of the following which are unsolicited input flows which activate the process, or are used to read and write data from a local data store:
(a) 'toll_tag_data_request' - which is a data flow received from the toll tag data collection process;
(b) 'toll_tag_data_needed' - which is a data flow received from the Manage Traffic function that contains a request for toll tag data;
(c) 'fpi-toll_tag_data' - which is s data flow received from the payment instrument terminator;
(d) 'toll_tag_data_store' - which reads/writes data from a local data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the first unsolicited input data flow in (a) is received, generate the output data flow containing the toll tag data;
(c) as a result of (b) await receipt of the data flows that either request toll payment from the credit stored on the payment instrument, or confirm that payment will be deducted from the credit identity supplied by the payment instrument, or contain an updated version of the toll tag data with the identity of the toll segment at which the vehicle entered the toll road;
(d) when either of the two flows in (c) is received, send the appropriate toll payment request or payment debited flow to the payment instrument;
(e) when the third flow in (c) is received, update the local store of toll tag data;
(f) when the second unsolicited data flow in (a) is received, send the data flow containing tag data to the Manage Traffic function;
(g) when the third unsolicited input flow in (a) is received, write the data to the local data store replacing any data that already exists in the store;
(h) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(i) use the most appropriate mechanism(s) such as RDBMS, to manage the data in the store identified above.

User Service Requirements:

USR = 3.0;

USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

driver_toll_payment_credit_identity = toll_tag_data_request;
toll_payment_confirmation = toll_tag_data_request;
toll_tag_data_collect = toll_tag_data_request;
toll_tag_data_input = toll_tag_data_needed;
toll_tag_data_store = toll_tag_data_request;
tpi-debited_payment_at_toll_plaza = toll_tag_data_request;
tpi-request_payment_at_toll_plaza = toll_tag_data_request;

7.2.1.1 Read Parking Lot Tag Data

Input Flows

parking_lot_tag_data_collect
vehicle_parking_lot_characteristic_data

Output Flows

get_parking_lot_tag_violator_image
parking_lot_tag_data_request
parking_lot_tag_data_update
parking_lot_tag_pull_in_message
vehicle_tag_for_charges
vehicle_type_for_charges

Description:

Overview: This process shall be responsible for requesting the data from the parking lot tag being carried on-board the vehicle and used as the payment instrument being read. If there is no tag or the data it contains cannot be properly read, the process shall send a message for the vehicle to pull in for output by another process, and send a request to other processes to obtain an image of the vehicle. If there is no entry time data on the tag, then the process shall re-write this data plus the number of the entry lane onto the tag, so that it can be used as the mechanism for charging for the use of the parking lot. If the entry time is present, the process shall combine it with the vehicle characteristics, e.g., size, type, etc. to form the data upon which the parking lot payment transaction can be based, and send it to another process.

Data Flows: All input and output data flows are solicited with the exception of the following which is used to trigger the process:
(a) 'vehicle_parking_lot_characteristic_data' - which is received from another process that detects a vehicle's presence.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

get_parking_lot_tag_violator_image = TAG_DEFECT_RATE*(vehicle_parking_lot_characteristic_data);
parking_lot_tag_data_request = vehicle_parking_lot_characteristic_data;
parking_lot_tag_data_update = vehicle_parking_lot_characteristic_data;
parking_lot_tag_pull_in_message = TAG_DEFECT_RATE*(vehicle_parking_lot_characteristic_data);
vehicle_tag_for_charges = vehicle_parking_lot_characteristic_data;
vehicle_type_for_charges = vehicle_parking_lot_characteristic_data;

7.2.1.10 Determine Advanced Charges

Input Flows

advanced_charges_needed
parking_lot_prices

Output Flows

advanced_parking_lot_billing

Description:

Overview: This process shall be responsible for receiving a request to pay an advanced parking lot charge. It shall obtain the required parking lot charge from a data store and shall then forward the data to the billing processes. The store of parking lot charges shall be maintained by another process.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'parking_lot_prices '.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(d) use the most appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.2;
USR = 3.1.4;
USR = 3.1.4.1;

Output Flow Dynamics Assumptions:

advanced_parking_lot_billing =
6/60*PARKING_LANES+10/60*TOLL_LANES*TOLL_PLAZAS+10/60*ITS_TRANSIT_VEHS;

7.2.1.2 Calculate Vehicle Parking Lot Charges

Input Flows

parking_lot_prices
vehicle_tag_for_charges
vehicle_type_for_charges

Output Flows

parking_lot_charge

Description:

Overview: This process shall be responsible for calculating the parking lot charge for the detected vehicle based on its characteristics and data obtained from the tag being carried by the vehicle. The process shall obtain the cost of the use of the parking lot by reading data from a store that contains the standard prices for parking lot charges.

exception of the following which contains data requested from a data store:
(a) 'parking_lot_prices'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) use the most appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.13;
USR = 1.8.2.13(c);
USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;
USR = 3.1.3.3;

Output Flow Dynamics Assumptions:

parking_lot_charge = 12/(60)*PARKING_LANES;

7.2.1.3 Collect Bad Charge Payment Data

Input Flows

charge_payment_violator_data
ffi_bad_charges_payment_updates

Output Flows

bad_charge_payment_list
tfi_parking_lot_payment_violator_data

Description:

Overview: This process shall be responsible for maintaining a data store containing a list of invalid driver credit identities. The process shall use this data to check credit identities provided for checking by the billing process. This checking shall ensure that the current parking lot payment transaction is using a credit identity that has not previously had an invalid transaction. Details of possible invalid credit identities shall be sent by the process to the financial institution for verification. The process shall also receive from the financial institution details of invalid payment instrument data that has been found by other means.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data written to a data store:
(a) 'bad_charge_payment_list'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) be responsible for the management of the data in the store of bad driver payment instrument data, using the most appropriate mechanism(s) such as RDBMS, for storing the data.
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

bad_payment_list = charge_payment_violator_data + ffi-bad_charges_payment_updates;
tfi-parking_lot_payment_violator_data = charge_payment_violator_data;

7.2.1.4 Check for Advanced Parking Lot Payment

Input Flows

advanced_charges_payment_list
advanced_parking_lot_billing
parking_lot_charge

Output Flows

advanced_charge_transactions
advanced_charges_payment_list
billing_for_charges_needed

Description:

Overview: This process shall be responsible for checking to see if the required parking lot charge payment has already been made. The process shall determine the existence of an advanced payment for the parking lot charges by comparing the received payment information with that in the store containing the list of advanced payments. If the payment has already been made then the process shall remove the requirement for local billing and remove the record of the advanced payment from the store. Details of each payment transaction shall be sent by the process to another process with the payment information received from the driver removed.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from and written to a data store:
(a) 'advanced_charges_payment_list'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) be responsible for the management of the data in the store of advanced charges payment data, using the most appropriate mechanism(s) such as RDBMS, for storing the data;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(e) remove the payment information received from the driver from all data that is sent to another process for loading into the store of parking lot charge payment transactions.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;
USR = 3.1.3.2;

Output Flow Dynamics Assumptions:

advanced_charge_transactions = 5/(60)*PARKING_LANES;
billing_for_charges_needed = 10/(60)*PARKING_LANES;

7.2.1.5 Bill Driver for Parking Lot Charges

Input Flows

bad_charge_payment_list
billing_for_charges_needed
parking_lot_payment_confirmation

Output Flows

advanced_charges_payment_list
charge_payment_violator_data
confirm_advanced_charges_payment
current_charge_transactions
get_charge_payment_violator_image
parking_lot_payment_debited
parking_lot_payment_pull_in_message
parking_lot_payment_request
parking_lot_tag_data_clear

Description:

Overview: This process shall be responsible for either obtaining payment for the current or advanced parking lot charge. The process shall achieve this either by requesting that the charge be deducted from the credit being stored by the parking lot tag that is acting as the payment instrument, or informing the driver that payment for the charge will be debited from the credit identity provided by the tag. Before sending data to the tag, the process shall check that either the credit identity is not already in the list of bad payers, or the stored credit is not less that the parking lot charge. If either of these conditions is true the process shall obtain an image of the driver and vehicle which can be forwarded to the appropriate enforcement agency via another process. When the appropriate payment transaction has been completed, the parking lot entry time data shall be cleared from the tag so that it can be used for the next visit by the vehicle to a parking lot. The tag may be in the form of some type of credit or debit card, or an electronic purse. Details of the transaction shall always be sent to the process that manages parking lot transactions which will also send details to the financial institution if a credit or debit card is involved. Where an advanced parking lot charge payment is identified, no action is taken if the credit identity is on the bad payers list, or the stored credit is less than the charge, other than the payment is not confirmed.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following:
(a) 'bad_charge_payment_list', which contains data requested from a data store;
(b) 'advanced_charges_payment_list', which contains data written to a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) when the unsolicited flow requesting billing for parking lot charges is received, check to see if it contains a credit identity or a stored credit value;
(d) if a credit identity is found in (c) that check it against those held in the store containing a list of bad charge payments;
(e) if a match is found in (d), get an image of the violator and output the data flow that requests the vehicle to pull in;
(f) if a stored credit value is found in (c) and it is less that the parking lot charge, get an image of the violator and output the data flow that requests the vehicle to pull in;
(g) if the stored credit value found in (c) is greater than or equal to the parking lot charge, send the output flow to the payment instrument requesting that the charge be deducted from the credit being stored by the instrument;
(h) if a negative response is received to (g), get an image of the violator and output the data flow that requests the vehicle to pull in;
(i) when the parking lot charge transaction is complete always output details of the transaction in the flow of current charge transactions and send out the data flow that clears the parking lot tag data store;
(j) if the payment is identified as being for an advanced charge, and a match is found in (d) or the stored credit value is less than the charge, then set the output flow of advanced parking lot charge payment to false and take no further action;
(k) if the tests if (j) are clear then set the output flow of advanced parking lot charge payment to true and enter the credit and vehicle identities into the store containing the advanced payment list;
(l) use the appropriate mechanism(s) such as RDBMS, to retrieve from and write data from to the

store identified above;
(m) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

advanced_charges_payment_list = 6/60*PARKING_LANES+10/60*TOLL_LANES*TOLL_PLAZAS+1/(60*60)*ITS_TRAVS;
charge_payment_violator_data = 8/(60*60*24)*PARKING_LANES;
confirm_advanced_charges_payment = 6/(60)*PARKING_LANES;
current_charge_transactions = 8/(60)*PARKING_LANES;
get_charge_payment_violator_image = 8/(60*60*24)*PARKING_LANES;
parking_lot_payment_debited = 8/(60)*PARKING_LANES;
parking_lot_payment_pull_in_message = 8/(60)*PARKING_LANES;
parking_lot_payment_request = 8/(60)*PARKING_LANES;
parking_lot_tag_data_clear = 8/(60)*PARKING_LANES;

7.2.1.6 Manage Parking Lot Financial Processing

Input Flows

advanced_charge_transactions
current_charge_transactions
ffi_confirm_charges_payment
fpo_transaction_reports_request
parking_lot_transaction_records

Output Flows

parking_lot_transaction_records
tfi_request_charges_payment
tpo_transaction_reports

Description:

Overview: This process shall be responsible for maintaining a log of all transactions that are carried out by other processes in the Process Electronic Parking Lot Payment facility. The identity of the payee shall have been removed from the data before it is stored. At periodic intervals the process shall output the accumulated records to another process in the Provide Electronic Payment Services function. It shall also output the same data on request to the parking operator, either in hardcopy form, or as a visual display. The process shall be responsible for sending details of transactions to the financial institution to enable the users to be billed through their credit identities. The input and output forms shall include those that are suitable for travelers with physical disabilities.

exception of the following which contains data written to and requested from a data store:
(a) 'parking_lot_transaction_records'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs of transaction records are received, load the data in the store of transaction records;
(c) when requested by the appropriate input, generate the outputs identified above;
(d) be responsible for the management of the data in the store of parking lot charge transaction records, using the most appropriate mechanism(s) such as RDBMS, for storing the data;
(e) remove all driver identities from the data before it is loaded into the data store;
(f) ensure that all output flows are encrypted in such a way that it is not possible to determine the financial data being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.8;
USR = 1.8.1;
USR = 1.8.1.6;
USR = 1.8.1.6(e);
USR = 1.8.2;
USR = 1.8.2.12;
USR = 1.8.2.12(b);
USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

parking_lot_transaction_records = advanced_charge_transactions + current_charge_transactions;
tfi-request_charges_payment = 1/(60*60*24)*TOLL_PLAZAS;
tpo-transaction_reports = fpo-transaction_reports_request;

7.2.1.7 Update Parking Lot Data

Input Flows

fpo_parking_lot_charge_change_response
fpo_parking_lot_data
other_parking_lot_price_data_request
parking_charge_request_for_archive
parking_lot_capacity_update_confirm
parking_lot_charge_change_request
parking_lot_charge_direct_request
parking_lot_price_data_request

Output Flows

other_parking_lot_price_data
parking_charge_response_for_archive
parking_lot_capacity_update
parking_lot_charge_change_response
parking_lot_charge_direct_details
parking_lot_price_data
parking_lot_prices
tpo_parking_lot_charge_change_request

Description:

Overview: This process shall be responsible for maintaining a store of data containing the parking lot charges, which may vary according to the type of vehicle. The process shall also act as the interface to the parking service provider to enable changes to be made to the stored data, for the output and input of responses to parking lot price change requests from the Manage Traffic function, and for requests for parking lot price data from the Centralized Payments facility. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'parking_lot_prices'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when any of the inputs from the parking operator are received, generate the appropriate output identified above;
(c) when the input for a parking lot prices change request is received from the Manage Transit function, generate the change request output to the parking service provider;
(d) when a request for parking lot price data is received, generate the data flow containing the copy of the store of price data;
(e) be responsible for the management of the data in the store of parking lot price data, using the appropriate mechanism(s) such as a relational database, for storing the data.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;
USR = 3.1.3.3;

Output Flow Dynamics Assumptions:

parking_lot_capacity_update = 1/(60*60*24*7*52)*PARKING_LOTS;
other_parking_lot_price_data = 1/(60*60*24*7*52)*PARKING_LOTS;
parking_lot_charge_change_response = parking_lot_charge_change_request;
parking_lot_charge_direct_details = parking_lot_charge_direct_request;
parking_lot_prices = fop-parking_lot_data;
parking_lot_price_data = parking_lot_price_data_request+fop-parking_lot_data;
tpo-parking_lot_charge_change_request = parking_lot_charge_change_request;
parking_charge_response_for_archive = parking_charge_request_for_archive;

7.2.1.8 Register for Advanced Parking Lot Payment

Input Flows

advanced_other_charges_request
advanced_traveler_charges_request
confirm_advanced_charges_payment
fpo_confirm_advanced_parking_payment
parking_lot_bookings_confirm

Output Flows

advanced_charges_needed
advanced_other_charges_confirm
advanced_traveler_charges_confirm
parking_lot_bookings_request
tpo_request_advanced_parking_payment

Description:

Overview: This process shall be responsible for responding to requests for parking lot charges to be paid in advance. It shall provide the parking operator with the opportunity to deny the request for advanced payment of a parking lot charge. If approved, the advanced parking lot charge data shall be forwarded by the process to other processes for the actual cost to be obtained and the payment transactions initiated.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs requesting advanced parking lot charges are received, generate the outputs to the parking operator identified above;
(c) if no response is received to the flows output in (b), assume that the advanced parking lot charges have been accepted and send the data to the advanced charge determination process;
(d) if a negative response is received to the flows in (b), then output the advanced parking lot charge response data flows with the confirmation data set to fail;
(e) when the confirm data flow is received from the bill driver for parking lot charges process, then output the advanced parking lot charge response data flows with the confirmation data set to true;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;
USR = 3.1.3.2;
USR = 3.1.4;
USR = 3.1.4.1;

Output Flow Dynamics Assumptions:

advanced_charges_needed = 6/60*PARKING_LANES+10/60*TOLL_LANES*TOLL_PLAZAS+1/(60*60)*ITS_TRAVS;
advanced_other_charges_confirm = 10/(60)*TOLL_LANES*TOLL_PLAZAS+1/(60*60)*ITS_TRAVS;
parking_lot_bookings_request = 6/60*PARKING_LANES+10/60*TOLL_LANES*TOLL_PLAZAS+1/(60*60)*ITS_TRAVS;
advanced_traveler_charges_confirm = 1/(60*60)*ITS_TRAVS;
tpo_request_advanced_parking_payment = 6/(60)*PARKING_LANES;

7.2.1.9 Manage Parking Lot Reservations

Input Flows

fpo_parking_lot_hours_of_operation
parking_lot_bookings_request
parking_lot_capacity_update
parking_lot_data
parking_lot_data_request
parking_lot_reservation_request

Output Flows

parking_lot_availability
parking_lot_bookings_confirm
parking_lot_capacity_update_confirm
parking_lot_data
parking_lot_reservation_confirm

Description:

Overview: This process shall be responsible for maintaining a store of parking lot data. This data shall cover the capacity of the parking lot, i.e., the maximum number of spaces available, which may vary according to the type of vehicle. The process shall also act as the interface for inquiries from other ITS functions both for details of parking lot capacity, both now and in the future and for the reservation of spaces as part of travelers' confirmed trips. The parking lot data also contains data on the hours of operation of parking lots. This data is used in transactions requiring electronic payment of parking lot services, as well as for a traveler making a parking lot reservation.

exception of the following which contains data requested from a data store:
(a) 'parking_lot_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) be responsible for the management of the data in the store of parking lot data, using the appropriate mechanism(s) such as RDBMS, for storing the data.
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.4;
USR = 3.1.4.3;

Output Flow Dynamics Assumptions:

parking_lot_availability = parking_lot_data_request;
parking_lot_bookings_confirm = parking_lot_bookings_request;
parking_lot_capacity_update_confirm = parking_lot_capacity_update;
parking_lot_data = parking_lot_capacity_update + parking_lot_reservation_request
+ parking_lot_bookings_request;
parking_lot_reservation_confirm = parking_lot_reservation_request;

7.2.2 Produce Parking Lot Displays

Input Flows

parking_lot_payment_pull_in_message
parking_lot_tag_pull_in_message

Output Flows

td_parking_lot_payment_confirmed
td_parking_lot_payment_invalid

Description:

Overview: This process shall be responsible for driving the displays that tell vehicles whether or not their parking lot charge payment has been confirmed or rejected. The process shall receive the data for output via the displays from other processes. The data input and output formats shall use an appropriate form of display that shall be easily readable under all lighting conditions and over the range of speeds that vehicles are expected to use when entering or leaving a parking lot. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

td-parking_lot_payment_confirmed = 32/(60)*PARKING_LANES;
td-parking_lot_payment_invalid = (1/(60*60)+16/(60*60*24))*PARKING_LANES;

7.2.3 Obtain Parking Lot Violator Image

Input Flows

From_Vehicle_Characteristics
get_charge_payment_violator_image
get_parking_lot_tag_violator_image

Output Flows

parking_lot_violation_information

Description:

Overview: This process shall be responsible for obtaining an image of a violator for use by other processes. The form of the image data obtained by this process shall be very accurate so that there can be no mistake of the determination of the identity of the vehicle and/or driver, and shall be easily passed on by the other processes to the appropriate law enforcement agency(ies) so that punitive action may be taken. The process shall be capable of obtaining an image of the required accuracy under all lighting conditions and over the range of speeds with which vehicles will enter or leave parking lots.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input data flows requesting a violator image be obtained are received, read the vehicle characteristics data being received;
(c) use the data in (b) to generate a highly accurate image of the vehicle and/or its driver;
(d) output the image in the violation information flow identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;

Output Flow Dynamics Assumptions:

parking_lot_violation_information = (1/(60*60)+16/(60*60*24))*PARKING_LANES;

7.2.4 Provide Driver Parking Lot Payment Interface

Input Flows

advanced_tolls_and_fares_response
driver_parking_payment_credit_identity
fbv_vehicle_identity
fd_other_services_parking_request

Output Flows

advanced_tolls_and_fares_request
driver_advanced_payment_at_lot
td_other_services_parking_response

Description:

Overview: This process shall be responsible for providing an interface through which drivers can request other services when paying their charges at parking lots. The services supported by this process include advanced parking lot payment, as well as advanced payment for tolls and transit fares. The process shall query the driver for sufficient information to enable the advanced toll, parking lot charge, and/or transit fare to be determined and the cost either billed to a credit identity provided by the driver's payment instrument, or deducted from credit stored on the instrument. The input and output forms shall include those that are suitable for travelers with physical disabilities.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the driver, the basic vehicle and the driver's payment credit identity input flows are received, check the content of the input data to ensure that there is sufficient data to enable the advanced parking lot charge, toll and/or transit fare to be determined;
(c) if the check in (b) is not confirmed, output an insufficient data to complete transaction message to the driver using the appropriate output flow, and highlighting the missing data;
(d) the data required in (b) about the advanced booking shall include such things as the identity of the parking lot plus the date and time for which a space is required, and/or the toll segments to be traversed, and/or the origin and destination of the transit route;
(e) the data in (b) must in addition to that for the advanced booking request, also include the vehicle identity and either the driver's credit identity or the credit, stored on the payment instrument being used by the driver;
(f) if the check in (e) is not confirmed, output an insufficient financial data (credit identity or stored credit) to complete transaction message to the driver using the appropriate output flow;
(g) if advanced toll payment is required output the appropriate flow to generate any required tolls only, using the new value of the stored credit;
(h) if advanced toll payment was required in (g), when a response is received again reduce the value of the stored credit;
(i) if an advanced transit fare is to be paid, output the appropriate flow to generate any advanced transit fares, using the new value of the stored credit generated by any advanced costs already produced from the above;
(j) when output and response to all the advanced booking requests are complete, total up all the costs and check that they are less than the original value of the stored credit and if not output an insufficient credit message to the driver using the appropriate output flow;
(k) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.4;
USR = 3.1.4.1;

Output Flow Dynamics Assumptions:

advanced_tolls_and_fares_request = fd-other_services_parking_request;
driver_advanced_payment_at_lot = fd-other_services_parking_request;
td-other_services_parking_response = fd-other_services_parking_request;

7.2.5 Detect Vehicle for Parking Lot Payment

Input Flows

From_Vehicle_Characteristics

Output Flows

vehicle_parking_lot_characteristic_data

Description:

Overview: This process shall be responsible for producing a vehicle's characteristics from data received by sensors located at or near the parking lot entry and exit lanes. The data shall be sent by the process to another process in a form suitable for use in calculating the parking lot charge for the vehicle. The process shall ensure that the data includes such things as vehicle size, type, identifiable features, etc.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.3;
USR = 3.1.3.1;
USR = 3.1.3.3;
USR = 3.1.4.4

Output Flow Dynamics Assumptions:

vehicle_parking_lot_characteristic_data = 12/(60)*PARKING_LANES;
;

7.2.6 Distribute Advanced Tolls and Fares

Input Flows

advanced_other_charges_confirm
advanced_tolls_and_fares_request
transfer_fares_to_charges
transfer_tolls_to_charges

Output Flows

advanced_other_charges_request
advanced_tolls_and_fares_response
transfer_charges_to_fares
transfer_charges_to_tolls

Description:

Overview: This process shall be responsible for receiving requests for advanced payment of parking lot charges from the toll or transit fare collection facilities within the Provide Electronic Payment Services function. It shall pass the requests on to another process in the Provide Electronic Parking Lot Payment facility, and shall return transaction success or failure details to the requesting process. The process shall also receive requests for the advanced payment of tolls and transit fares from the parking lot payment interface process. It shall send these requests to other processes in the Provide Electronic Payment Services function and when received, return the results to the Parking Lot payment interface process.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs from other processes in the Provide Electronic Payment Services function are received, generate the advanced parking lot charge request output identified above;
(c) when a response is received to the flow in (b), return it in the flow to the requesting process in the Provide Electronic Payment Services function;
(d) when the inputs requesting advanced toll or transit fare payment are received from the parking lot payment interface process, generate the appropriate transfer flows to other processes in the Provide Electronic Payment Services function;
(e) when a response is received to the flows in (d), return it in the flow to the requesting process;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.1;
USR = 3.1.1.3;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.4;
USR = 3.1.4;
USR = 3.1.4.1;

Output Flow Dynamics Assumptions:

advanced_other_charges_request = 10/(60)*TOLL_LANES*TOLL_PLAZAS+10/(60)*ITS_TRANSIT_VEHS;
advanced_tolls_and_fares_response = 4/(60)*PARKING_LANES*PARKING_LOTS;
transfer_charges_to_fares = 4/(60)*PARKING_LANES*PARKING_LOTS;

7.2.7 Provide Payment Instrument Interface for Parking

Input Flows

driver_advanced_payment_at_lot
fpi_confirm_payment_at_parking_lot
fpi_parking_tag_data
parking_lot_payment_debited
parking_lot_payment_request
parking_lot_tag_data_clear
parking_lot_tag_data_needed
parking_lot_tag_data_read
parking_lot_tag_data_request
parking_lot_tag_data_update

Output Flows

driver_parking_payment_credit_identity
parking_lot_payment_confirmation
parking_lot_tag_data_collect
parking_lot_tag_data_input
parking_lot_tag_data_write
tpi_debited_payment_at_parking_lot
tpi_request_payment_at_parking_lot

Description:

Overview: This process shall be responsible for providing the interface through which the payment information can be read from a vehicle tag. The process shall enable the use of the data from the tag for the purposes of paying the current parking lot charge and if required, advanced payments for tolls and/or transit fares. It shall be possible for the process to collect either the credit identity or the stored credit value data from the tag, and to update the stored credit value as a result of the parking lot charge and (possibly) advanced charges having been paid. The time at which the vehicle entered the parking lot shall also be collected from the tag by the process so that the charge for the use of the lot can be calculated. The process shall support collection of data from tags on-board a range of vehicle types including private cars or vans, commercial vehicles, transit vehicles, including those used for demand responsive transit services.

Data Flows: All input and output data flows are solicited with the exception of the following which are unsolicited input flows which activates the process, or are used to read and write data from a local data store:
(a) 'parking_lot_tag_data_request' - which is a data flow received from the parking lot tag data collection process;
(b) 'parking_lot_tag_data_needed' - which is a data flow received from the Manage Traffic function that contains a request for toll tag data;
(c) 'fpi-parking_tag_data' - which is a data flow received from the payment instrument terminator;
(d) 'parking_lot_tag_data_read' - which reads data from a local data store;
(e) 'parking_lot_tag_data_write' - which writes data to a local data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the first unsolicited input data flow in (s) is received, generate the output data flow containing parking lot tag data;
(c) as a result of (b) await receipt of the data flows that either request parking lot charge payment from the credit stored on the payment instrument, or confirm that payment will be deducted from the credit identity supplied by the payment instrument, or contain an updated version of the parking lot tag data with the time at which the vehicle entered the parking lot and the lot entry lane number;
(d) when either of the first two flows in (c) is received, send the appropriate parking lot charge payment request or payment debited flow to the payment instrument;
(e) when the third flow in (c) is received, update the local store of parking lot tag data;
(f) when the second unsolicited data flow in (a) is received, send the data flow containing tag data to the Manage Traffic function;
(g) when the third unsolicited input flow in (a) is received, write the data to the local data store replacing any data that already exists in the store;
(h) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(i) use the most appropriate mechanism(s) such as RDBMS, to manage the data in the store identified above.

User Service Requirements:

USR = 3.0;

USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

driver_parking_payment_credit_identity = parking_lot_tag_data_request;
parking_lot_payment_confirmation = parking_lot_tag_data_request;
parking_lot_tag_data_collect = parking_lot_tag_data_request;
parking_lot_tag_data_input = parking_lot_tag_data_needed;
parking_lot_tag_data_write = parking_lot_tag_data_request;
tpi-debited_payment_at_parking_lot = parking_lot_tag_data_request;
tpi-request_payment_at_parking_lot = parking_lot_tag_data_request;

7.3.1.1 Register for Advanced Transit Fare Payment

Input Flows

advanced_other_fares_request
advanced_traveler_fares_request
confirm_advanced_fares_payment

Output Flows

advanced_fares_needed
advanced_other_fares_confirm
advanced_traveler_fares_confirm

Description:

Overview: This process shall be responsible for responding to requests for transit fares to be paid in advance. The advanced transit fare data shall be forwarded by the process to other processes for the actual cost to be obtained and the payment transactions initiated.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.4;
USR = 3.1.2.8;

Output Flow Dynamics Assumptions:

advanced_traveler_fares_confirm = 1/3600*ITS_TRAVS;
advanced_fares_needed =
1/3600*ITS_TRAVS+1/6*TOLL_LANES*TOLL_PLAZAS+1/15*PARKING_LANES*PARKING_LOTS;
advenced_fare_payment_list =
1/3600*ITS_TRAVS+1/6*TOLL_LANES*TOLL_PLAZAS+1/15*PARKING_LANES*PARKING_LOTS;
advanced_other_fares_confirm =
1/6*TOLL_LANES*TOLL_PLAZAS+1/15*PARKING_LANES*PARKING_LOTS;

7.3.1.2 Determine Advanced Transit Fares

Input Flows

advanced_fares_needed
transit_fares_for_advanced_payments
transit_services_for_advanced_fares

Output Flows

advanced_fare_billing

Description:

Overview: This process shall be responsible for receiving a request to pay an advanced transit fare. It shall obtain the required transit fare data from a local store of transit fares and shall then forward the data to the billing processes. The store of fare data shall be maintained by another process.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'transit_fares_for_advanced_payments'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(d) use the most appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.2;
USR = 3.1.2.8;

Output Flow Dynamics Assumptions:

advanced_fare_billing =
1/3600*ITS_TRAVS+1/6*TOLL_LANES*TOLL_PLAZAS+1/15*PARKING_LANES*PARKING_LOTS;

7.3.1.3 Manage Transit Fare Financial Processing

Input Flows

advanced_fare_transactions
current_fare_transactions
ffi_confirm_fare_payment
transit_fare_transaction_records

Output Flows

tfi_request_fare_payment
transit_fare_transaction_records
transit_fare_transactions
ttfm_transaction_reports
ttso_transaction_reports

Description:

Overview: This process shall be responsible for maintaining a log of all the transactions carried out by other processes in the Process Electronic Transit Fare Payment facility. The identity of the payee shall have been removed from the data before it is stored. At periodic intervals the process shall output the accumulated records to the transit fleet manager, the transit system operator and to another process in the Provide Electronic Payment Services function. The process shall also be responsible for sending details of transactions to the financial institution to enable the users to be billed through their credit identities. The input and output forms shall include those that are suitable for travelers with physical disabilities.

exception of the following which contains data written to and requested from a data store:
(a) 'transit_fare_transaction_records'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, load the data in the store of transaction records;
(c) at periodic intervals, e.g. daily, generate the outputs identified above;
(d) be responsible for the management of the data in the store of transit fare transaction records, using the appropriate mechanism(s) such as RDBMS, for storing the data;
(e) remove all transit user identities from the data before it is loaded into the data store;
(f) ensure that all output flows are encrypted in such a way that it is not possible to determine the financial data being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.8.2;
USR = 1.8.2.10;
USR = 1.8.2.10(a);
USR = 2.3.0;
USR = 2.3.3;
USR = 2.3.3.1;
USR = 2.3.3.1(c);
USR = 2.3.3.3;
USR = 3.0;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.3;
USR = 3.1.2.5;

Output Flow Dynamics Assumptions:

tfi-request_fare_payment = 1/(60*60*24);
transit_fare_transaction_records = advanced_fare_transactions + current_fare_transactions;
ttfm-transaction_reports = 1/(60*60*24);
ttso-transaction_reports = 1/(60*60*24);
transit_fare_transactions = 1/(60*60*24)

7.3.1.4 Check for Advanced Transit Fare Payment

Input Flows

advanced_fare_billing
advanced_fare_payment_list
request_roadside_fare_payment
request_vehicle_fare_payment

Output Flows

advanced_fare_transactions
billing_for_fares_needed

Description:

Overview: This process shall be responsible for checking to see if the required transit fare payment has already been made. The process shall determine the existence of an advance payment for

the transit fare by comparing the received payment information with that in the store containing the list of advanced payments. If the payment has already been made then the process shall remove the requirement for local billing and remove the record of the advanced payment from the store. Details of each payment transaction shall be sent by the process to another process with the payment information received from the transit user removed.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'advanced_fare_payment_list'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) use the most appropriate mechanism(s) such as RDBMS, to retrieve data from the store identified above;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques;
(e) remove the credit identity of the transit user from all data that is sent to another process for loading into the store of transit fare payment transactions.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.4;
USR = 3.1.2.8;
USR = 3.1.4;

Output Flow Dynamics Assumptions:

advanced_fare_transactions = advanced_fare_billing;
billing_for_fares_needed =
1/3600*ITS_TRAVS+1/6*TOLL_LANES*TOLL_PLAZAS+1/15*PARKING_LANES*PARKING_LOTS;

7.3.1.5 Bill Transit User for Transit Fare

Input Flows

bad_fare_payment_list
bad_tag_list_request
billing_for_fares_needed
transit_roadside_fare_payment_confirmation
transit_vehicle_fare_payment_confirmation

Output Flows

advanced_fare_payment_list
bad_tag_list_update
confirm_advanced_fares_payment
confirm_roadside_fare_payment
confirm_vehicle_fare_payment
current_fare_transactions
fare_payment_violator_data
get_fare_violator_payment_image
transit_roadside_fare_payment_debited
transit_roadside_fare_payment_request
transit_vehicle_fare_payment_debited
transit_vehicle_fare_payment_request

Description:

Overview: This process shall be responsible for obtaining payment for a transit fare transaction using data provided by the transit user. The process shall achieve this either by requesting that the fare be deducted from the credit being stored by the tag that is acting as the payment instrument for the transit user, or informing the transit user that payment for the fare will be debited to the credit identity provided by the tag. Before sending data to the tag, the process shall check that the transit user's credit identity is not already in the list of bad payers, and if it is request an image of the user which can be forwarded to the appropriate enforcement agency via another process. The tag may be in the form of cash, some type of credit or debit card, an electronic purse, or an intelligent transit ticket upon which pre-payment has been recorded, etc. Details of the transaction shall always be sent by the process to the process that manages transit fare transactions. The process shall load details of advanced transit fare payments into a data store for use by another process when the transit user eventually passes a fare payment point. If requested the process shall provide a copy of the current bad payers list to processes in the transit vehicle fare collection facility for use in on-board payment validation.

exception of the following which contains data requested from a data store:
(a) 'bad_fare_payment_list', which contains data requested from a data store;
(b) 'advanced_fare_payment_list', which contains data written to a data store.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above that contain data about fare billing that is needed, or is a request for a copy of the bad payers list;
(b) when the data about fare billing is received, check any credit identity that it contains against the list of bad payers;
(c) if a match is found in (b) then output the data flow requesting a violator image and return a fare payment failed message to the source of the billing information;
(d) if no match is found in (b) then either request that the tag's stored credit be reduced by the amount of the fare, or send the payment information to another process for debiting via the financial institution;
(e) as a result of (d), send a fare payment successful message to the source of the billing information;
(f) if the deduction of the fare from the tag's stored credit in (d) fails then output the data flow requesting a violator image and return a fare payment failed message to the source of the billing information;
(g) if in (d) the payment information is sent to another process for output to the financial institution, then output a fare payment debited message to the source of the billing information;
(h) if the billing information was for the advanced payment of a transit fare, then on successful completion of the payment transaction in the previous steps, store the data about the source of the payment and the fare to which it applies in the store of advanced payments;
(i) when the data flow received in (a) is a request for a copy of the bad payers list, read the current list from the data store and send it to the requesting process;
(j) use the most appropriate mechanism(s) such as RDBMS, to retrieve and write data from and to the stores identified above;
(k) all input and output flows must be encrypted in such a way that it is not possible to determine

the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.3;
USR = 3.1.2.8;

Output Flow Dynamics Assumptions:

bad_tag_list_update = bad_tag_list_request;
confirm_advanced_fares_payment = 1/3600*ITS_TRAVS+1/6*TOLL_LANES*TOLL_PLAZAS
+1/15*PARKING_LANES*PARKING_LOTS;
confirm_roadside_fare_payment = request_roadside_fare_payment;
confirm_vehicle_fare_payment = request_vehicle_fare_payment;
current_fare_transactions = billing_for_fares_needed;
fare_payment_violator_data = 1/(60*60)*ITS_TRANSIT_VEHS;
get_fare_violator_payment_image = 8/(60*60)*ITS_TRANSIT_VEHS;
tpi-request_fare_payment =
1*ITS_TRANSIT_VEHS+1/3600*ITS_TRAVS+1/6*TOLL_LANES*TOLL_PLAZAS
+1/15*PARKING_LANES*PARKING_LOTS;
transit_roadside_fare_payment_debited = request_roadside_fare_payment;
transit_roadside_fare_payment_request = request_roadside_fare_payment;
transit_vehicle_fare_payment_debited = request_vehicle_fare_payment;
transit_vehicle_fare_payment_request = request_vehicle_fare_payment;

7.3.1.6 Collect Bad Transit Fare Payment Data

Input Flows

fare_payment_violator_data
ffi_bad_fare_payment_updates

Output Flows

bad_fare_payment_list
tfi_fare_payment_violator_data

Description:

Overview: This process shall be responsible for maintaining a data store containing a list of invalid transit user credit identities. The process shall use this data to check credit identities provided for checking by the billing process. This checking shall ensure that the current transit fare payment transaction is using a credit identity that has not previously had an invalid transaction. Details of possible invalid credit identities shall be sent by the process to the financial institution for verification. The process shall also receive from the financial institution details of invalid payment instrument data that has been found by other means.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data written to a data store:
(a) 'bad_fare_payment_list'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the inputs are received, generate the outputs identified above;
(c) be responsible for the management of the data in the store of bad transit user payment instrument data, using the most appropriate mechanism(s) such as RDBMS, for storing the data.
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.2.3;
USR = 3.1.2.5;

Output Flow Dynamics Assumptions:

bad_fare_payment_list = fare_payment_violator_data;
tfi-fare_payment_violator_data = fare_payment_violator_data;

7.3.1.7 Update Transit Fare Data

Input Flows

ftso_fare_updates
ftso_request_fare_output
transit_fare_data_request
transit_fare_direct_request

Output Flows

transit_fare_data
transit_fare_direct_details
transit_fares_for_advanced_payments
transit_roadside_fare_data
transit_vehicle_fare_data
ttso_transit_fare_output

Description:

Overview: This process shall be responsible for managing the store of data that provides the actual value of transit fares for each segment of each regular transit route. The process shall also act as the interface through which the transit system operator can output and make changes to the stored data, and copies of this data can be provided to the Centralized Payments facility on request. The process shall support inputs from the transit system operator in both manual and audio form, and shall provide its outputs in audible and visual forms. It shall enable the visual output to be either in hardcopy, or as a display. The process shall automatically output the new fares for use by process on-board a transit vehicle and at the roadside, as well as by other ITS functions.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data written to a data store:
(a) 'transit_fares_for_advanced_payments'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows from the transit system operator listed above;
(b) when the input requesting output of the current data is received, generate the output of fares to the operator using the data flow identified above;
(c) when the input containing new fare data is received, load the new data into the data store, and generate the other output data flows identified above.
(d) when a request for transit fare price data is received, generate the data flow containing the copy of the store of price data;
(e) use the most appropriate mechanism(s) such as a relational database, to write data to the store of transit fares identified above.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.6;

Output Flow Dynamics Assumptions:

transit_vehicle_fare_data = ftso-fare_updates;
transit_roadside_fare_data = ftso-fare_updates;
transit_fare_data = transit_fare_data_request+ftso-fare_updates;
transit_fare_direct_details = transit_fare_direct_request
transit_fares_for_advanced_payments = ftso-fare_updates;
ttso-transit_fare_output = ftso-request_fare_output;

7.3.2 Distribute Advanced Tolls and Parking Lot Charges

Input Flows

advanced_other_fares_confirm
advanced_tolls_and_charges_roadside_request
advanced_tolls_and_charges_vehicle_request
transfer_charges_to_fares
transfer_tolls_to_fares

Output Flows

advanced_other_fares_request
advanced_tolls_and_charges_roadside_confirm
advanced_tolls_and_charges_vehicle_confirm
transfer_fares_to_charges
transfer_fares_to_tolls

Description:

Overview: This process shall be responsible for receiving requests for advanced payment of transit fares from the toll and parking lot charge collection facilities within the Provide Electronic Payment Services function. It shall pass the advanced fare requests on to another process in the Process Electronic Transit Fare Payment facility, and when received, shall return transit success or failure details to the requesting process. The process shall also receive requests for advanced payment of tolls and parking lot charges from transit vehicle and roadside (transit stop) fare collection facilities. It shall send these requests to other processes in the Provide Electronic Payment Services function and when received, return the results to the requesting process.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when the inputs from other processes in the Provide Electronic Payment Services function are received, generate the advanced fare request output identified above;
(c) when a response is received to the flow in (b), return it in the flow to the requesting process in the Provide Electronic Payment Services function;
(d) when the inputs requesting advanced toll or parking lot charge payment are received, generate the appropriate transfer flows to other processes in the Provide Electronic Payment Services function;
(e) when a response is received to the flows in (d), return it in the flow to the requesting process in the transit vehicle or roadside;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.1;
USR = 3.1.4;
USR = 3.1.4.1;
USR = 3.1.4.2;

Output Flow Dynamics Assumptions:

advanced_other_fares_request =
4/60*PARKING_LANES*PARKING_LOTS+10/60*TOLL_LANES*TOLL_PLAZAS;

7.3.3 Get Transit User Image for Violation

Input Flows

get_fare_violator_payment_image
transit_user_roadside_image
transit_user_vehicle_image

Output Flows

fare_violation_information
request_transit_user_roadside_image
request_transit_user_vehicle_image

Description:

Overview: This process shall be responsible for obtaining an image of a transit user who is trying to carry out an invalid fare payment transaction. The process shall send the image request to other processes either at the roadside, i.e., a transit stop, or on-board a transit vehicle, depending on where the transaction is being attempted. However if the collection method is set to batch, then the process shall take no further action, as an image of the offending transit user will not be available. When the image is received, the process shall use it to form part of the data sent to a process in the Manage Emergency Services function for forwarding to the appropriate enforcement agency.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input flows listed above;
(b) when the input requesting a violator image is received, check to see if the roadside or vehicle identity is set, i.e. non-zero;
(c) generate the output to the vehicle or roadside using the data flows identified above depending on which identity is set;
(d) when the input corresponding to the output in (b) is received, generate the fare violation information output identified above to the process in the Manage Emergency Services function;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.
advanced_tolls_and_charges_roadside_confirm = advanced_tolls_and_charges_roadside_request;
advanced_tolls_and_charges_vehicle_confirm = advanced_tolls_and_charges_vehicle_request;
transfer_fares_to_charges = 10/60*ITS_TRANSIT_VEHS;

User Service Requirements: transfer_fares_to_tolls = 10/60*ITS_TRANSIT_VEHS;

USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.5;

Output Flow Dynamics Assumptions:

fare_violation_information = (8/(60*60)+8/(60*60*24*7*52))*ITS_TRANSIT_VEHS
+(8/(60*60)+8/(60*60*24*7*52))*TRANSIT_STOPS;
request_transit_user_roadside_image = (8/(60*60)+8/(60*60*24*7*52))*TRANSIT_STOPS;
request_transit_user_vehicle_image = (8/(60*60)+8/(60*60*24*7*52))*ITS_TRANSIT_VEHS;

7.3.4 Provide Remote Terminal Payment Instrument Interface

Input Flows

fpi_confirm_fare_payment_at_roadside
fpi_transit_roadside_tag_data
transit_roadside_fare_payment_debited
transit_roadside_fare_payment_request

Output Flows

tpi_debited_fare_payment_at_roadside
tpi_request_fare_payment_at_roadside
transit_roadside_fare_payment_confirmation
transit_user_roadside_tag_data

Description:

Overview: This process shall be responsible for providing the interface through which payment information can be read from a transit user tag. The process shall support reading this data from transit users at the roadside, e.g., a transit stop, for use in paying the current transit fare and (if required) advanced payments. The process shall support advanced payments for tolls, parking lot charges, and/or transit fares. The process shall collect either the credit identity or the stored credit value data from the tag, and update the stored credit value as a result of the fare and (possibly) advanced charges.

Data Flows: All input and output data flows are solicited with the exception of the following which is an unsolicited input flow and which activates the process:
(a) 'fpi-transit_roadside_tag_data' - which is a flow received from the payment instrument terminator.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the output data flow containing transit roadside tag data;
(c) as a result of (b) await receipt of the data flows that either request transit fare payment from the credit stored on the payment instrument or confirm that payment will be deducted from the credit identity supplied by the payment instrument;
(d) when either of the flows in (c) is received, send the appropriate transit fare payment request or payment debited flow to the payment instrument;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog technique.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

transit_roadside_fare_payment_confirmation = fpi-transit_roadside_tag_data;
transit_user_roadside_tag_data = fpi-transit_roadside_tag_data;
tpi-debited_fare_payment_at_roadside = fpi-transit_roadside_tag_data;
tpi-request_fare_payment_at_roadside = fpi-transit_roadside_tag_data;

7.3.5 Provide Transit Vehicle Payment Instrument Interface

Input Flows

fpi_confirm_fare_payment_on_transit_vehicle
fpi_transit_vehicle_tag_data
transit_vehicle_fare_payment_debited
transit_vehicle_fare_payment_request

Output Flows

tpi_debited_payment_on_transit_vehicle
tpi_request_fare_payment_on_transit_vehicle
transit_user_vehicle_tag_data
transit_vehicle_fare_payment_confirmation

Description:

Overview: This process shall be responsible for providing the interface through which the payment information can be read from a transit user tag. The process shall support the reading of this data from transit users embarking on-board transit vehicles, for use in paying the current transit fare, and if required, advanced payments. The process shall support advanced payments for tolls, and/or parking lot charges, and/or transit fares. It shall be possible for the process to collect either the credit identity or the stored credit value data from the tag, and to update the stored credit value as a result of the fare and (possibly) advanced charges having been paid.

Data Flows: All input and output data flows are solicited with the exception of the following which is an unsolicited input flow and which activates the process:
(a) 'fpi-transit_vehicle_tag_data' - which is a flow received from the payment instrument terminator.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the output data flow containing transit vehicle tag data;
(c) as a result of (b) await receipt of the data flows that either request transit fare payment from the credit stored on the payment instrument or confirm that payment will be deducted from the credit identity supplied by the payment instrument;
(d) when either of the flows in (c) is received, send the appropriate transit fare payment request or payment debited flow to the payment instrument;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the credit identity or stored credit value being transmitted using any form of digital or analogue technique.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

transit_vehicle_fare_payment_confirmation = fpi-transit_vehicle_tag_data;
transit_user_vehicle_tag_data = fpi-transit_vehicle_tag_data;
tpi-debited_payment_on_transit_vehicle = fpi-transit_vehicle_tag_data;
tpi-request_fare_payment_on_transit_vehicle = fpi-transit_vehicle_tag_data;

7.4.1.1 Process Commercial Vehicle Payments

Input Flows

ffi_cv_payment_confirm
financial_request

Output Flows

financial_response
tfi_cv_payment_request

Description:

Overview: This process shall be responsible for transacting payments for electronic credential and tax filing by processes in the Manage Commercial Vehicles function. The payment transaction shall be initiated by processes in the Administer Commercial Vehicles facility which may accept inputs from both the commercial vehicle fleet manager and the commercial vehicle driver acting in the role of fleet manager, i.e., the owner driver. The process shall send the transaction data to the financial institution and report the response back to the requesting process.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when the input is received from the Manage Commercial Vehicles function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.4;
USR = 1.4.0;
USR = 1.4.2;
USR = 1.4.2.1;
USR = 1.4.2.4;
USR = 1.4.3;
USR = 1.4.3.2;
USR = 3.0;
USR = 3.1;
USR = 3.1.4;
USR = 3.1.4.1;

Output Flow Dynamics Assumptions:

financial_response = financial_request;
tfi-cv_payment_request = financial_request;

7.4.1.2 Process Yellow Pages Services Provider Payments

Input Flows

ffi_registration_payment_confirm
yellow_pages_service_provider_registration_request

Output Flows

tfi_registration_payment_request
yellow_pages_provder_payments_transactions
yellow_pages_service_provider_registration_response

Description:

Overview: This process shall be responsible for transacting payments for the registration of other (yellow pages) service providers. The process shall be initiated by receiving data from a process in the Provide Driver and Traveler Services function and shall send the data to the financial institution. The process shall send the response from the financial institution to the requesting process and shall send details of the transaction to another process for entry into a store of transaction records.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input data flows listed above;
(b) when the input is received from the Provide Driver and Traveler Services function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 2.0;
USR = 2.2;
USR = 2.2.1;
USR = 2.2.1.1;
USR = 2.2.1.1.4;
USR = 2.3;
USR = 2.3.4;
USR = 4.0;
USR = 4.1;
USR = 4.1.0;
USR = 4.1.2;
USR = 4.1.2.2;

Output Flow Dynamics Assumptions:

yellow_pages_service_provider_registration_response =
yellow_pages_service_provider_registration_request;
tfi-registration_payment_request = yellow_pages_service_provider_registration_request;
yellow_pages_provder_payments_transactions = yellow_pages_service_provider_registration_request;

7.4.1.3 Process Driver Map Update Payments

Input Flows

driver_map_update_payment_request
ffi_driver_map_payment_confirm

Output Flows

driver_map_update_payment_response
driver_map_update_payments_transactions
tfi_driver_map_payment_request

Description:

Overview: This process shall be responsible for transacting payments from the driver for updates to the navigable map database in the vehicle. The process shall receive the transaction request data from a process in the Provide Driver and Traveler Services function and shall send the data to the financial institution for action. The process shall send the response from the financial institution to the requesting process and shall send details of the transaction to another process for entry into the payment_transaction_records data store.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received from the Provide Driver and Traveler Services function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.3;
USR = 1.3.0;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.1;
USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.4;

Output Flow Dynamics Assumptions:

driver_map_update_payment_response = driver_map_update_payment_request;
driver_map_update_payments_transactions = driver_map_update_payment_request;
tfi-driver_map_payment_request = driver_map_update_payment_request;

7.4.1.4 Process Traveler Map Update Payments

Input Flows

ffi_traveler_display_payment_confirm
ffi_traveler_map_payment_confirm
traveler_map_update_payment_request
traveler_personal_display_update_payment_request

Output Flows

tfi_traveler_display_payment_request
tfi_traveler_map_payment_request
traveler_map_update_payment_response
traveler_map_update_payments_transactions
traveler_personal_display_update_payment_response

Description:

Overview: This process shall be responsible for transacting payments from the traveler for updates to the navigable map database carried in the personal device. The process shall receive the transaction request data from a process in the Provide Driver and Traveler Services function and shall send the data to the financial institution. The process shall send the response from the financial institution to the requesting process and shall send details of the transaction to another process for entry into the payment_transaction_records data store.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received from the Provide Driver and Traveler Services function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.3;
USR = 1.3.0;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.1;

Output Flow Dynamics Assumptions:

tfi-traveler_display_payment_request = traveler_personal_display_update_payment_request;
tfi-traveler_map_payment_request = traveler_map_update_payment_request;
traveler_map_update_payment_response = traveler_map_update_payment_request;
traveler_map_update_payments_transactions = traveler_map_update_payment_request;
traveler_personal_display_update_payment_response =

7.4.1.5 Process Transit User Other Services Payments

Input Flows

ffi_other_services_payment_confirm
other_services_roadside_request
other_services_vehicle_request

Output Flows

other_services_roadside_response
other_services_vehicle_response
tfi_other_services_payment_request
transit_user_payments_transactions

Description:

Overview: This process shall be responsible for collecting advance payments for other (yellow pages) services. The transaction data shall be provided by processes in the Manage Transit function in response to reservation requests from a transit user either at the roadside, i.e., a transit stop, or on-board a transit vehicle. The process shall send the received transaction data to the financial institution and shall send the response to the requesting process. It shall also send details of the transaction to another process for entry into a store of transaction records.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received from the Manage Transit function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.3;
USR = 1.3.0;
USR = 1.3.3;
USR = 1.3.3.2;
USR = 1.3.3.2.1;
USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.4;

Output Flow Dynamics Assumptions:

other_services_roadside_response = other_services_roadside_request;
other_services_vehicle_response = other_services_vehicle_request;
tfi-other_services_payment_request = other_services_roadside_request+other_services_vehicle_request;
transit_user_payments_transactions = other_services_roadside_request+other_services_vehicle_request;

7.4.1.6 Process Traveler Trip and Other Services Payments

Input Flows

ffi_traveler_other_services_payments_confirm
traveler_advanced_payments_confirm
traveler_other_services_payment_request
traveler_payment_request

Output Flows

tfi_traveler_other_services_payments_request
traveler_advanced_payments_request
traveler_other_services_payment_result
traveler_payment_response
traveler_trip_payments_transactions

Description:

Overview: This process shall be responsible for transacting advanced payments required for the confirmation of a trip by a traveler. Payments supported by the process shall comprise those for any tolls, parking lot charges, transit fares, or other (yellow pages) services that need to be paid for the trip to be confirmed. The process shall receive the transaction data from a process in the Provide Driver and Traveler Services function and shall send the data to the financial institution. Tolls, fares and parking lot charges are sent to the Route Traveler Advanced Payment function for processing. The process shall send the response from the financial institution to the requesting process and shall send details of the transaction to another process for entry into the payment_transaction_records data store.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received from the Provide Driver and Traveler Services function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.4;

Output Flow Dynamics Assumptions:

tfi-traveler_other_services_payments_request = traveler_payment_request;
traveler_advanced_payments_request = traveler_payment_request;
traveler_other_services_payment_result = traveler_other_services_payment_request;
traveler_payment_response = traveler_payment_request;
traveler_trip_payments_transactions =

7.4.1.7 Collect Payment Transaction Records

Input Flows

driver_map_update_payments_transactions
payment_transaction_records
traveler_map_update_payments_transactions
traveler_rideshare_payments_transactions
traveler_trip_payments_transactions
yellow_pages_provder_payments_transactions

Output Flows

payment_transaction_records
traveler_info_payments_transactions

Description:

Overview: This process shall be responsible for the collection and maintenance of a data store that contains transaction records for payments made for various services provided. The process shall load information into the payment_transaction_records data store for services comprising updates of map databases for drivers and travelers, registration of other (yellow pages) service providers (so that information about what they have to offer is available to travelers and transit users), advanced payment of tolls, parking lot charges, transit fares and other (yellow pages) services that form part of travelers' trips. The data shall be stored by the process with all references to the identity of the payment source, i.e., driver, traveler, commercial vehicle fleet manager, and any other payment information, removed.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data requested from a data store:
(a) 'payment_transaction_records'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received, generate the outputs identified above
(c) be responsible for the management of the data in the store of payment transactions, using the most appropriate mechanism(s) such as RDBMS, for storing the data;
(d) ensure that all references to the identity of those making the payments to which the transaction records relate are removed from the data before it is loaded into the store identified above;
(e) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.4;
USR = 3.1.4.1;

Output Flow Dynamics Assumptions:

traveler_info_payments_transactions = 1/(60*60);
payment_transaction_reports = 1/(60*60*24);

7.4.1.8 Process Traveler Rideshare Payments

Input Flows

ffi_traveler_rideshare_payment_confirm
rideshare_payment_request

Output Flows

rideshare_payment_confirmation
tfi_traveler_rideshare_payment_request
traveler_rideshare_payments_transactions

Description:

Overview: This process shall be responsible for transacting payments for ridesharing that are required for the confirmation of a traveler's trip. The process shall start the transaction by receiving data from a process in the Provide Driver and Traveler Services function and shall send the data to the appropriate financial institution. The process shall send the response from the financial institution to the requesting process and shall send details of the transaction to another process for entry into a store of transaction records.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input is received from the Provide Driver and Traveler Services function, generate the output to the financial institution identified above;
(c) as a result of (b) await the response from the financial institution and when received, use it to generate the reply to the requesting process;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.2;

Output Flow Dynamics Assumptions:

rideshare_payment_confirmation = rideshare_payment_request;
tfi-traveler_rideshare_payment_request = rideshare_payment_request;
traveler_rideshare_payments_transactions = rideshare_payment_request;

7.4.2 Collect Price Data for ITS Use

Input Flows

parking_lot_charge_request
parking_lot_price_data
price_data_for_services
request_prices
toll_price_data
toll_price_request
transit_fare_data
transit_fare_request

Output Flows

parking_lot_charge_details
parking_lot_price_data_request
price_data_for_services
prices
toll_price_data_request
toll_price_details
transit_fare_data_request
transit_fare_details

Description:

Overview: This process shall be responsible for collecting data about the prices being charged for tolls, parking lots and transit fares. This process shall accept data sent to it by the other processes when they have updated their data and automatically sent it, or this process shall request a transfer of data from the other processes. The process shall load the data into the price_data_for_services data store from which some or all of it can be read on request from processes in other ITS functions.

Data Flows: All input data flows are unsolicited and all output flows are solicited with the exception of the following which contains data written to and requested from a data store:
(a) 'financial_and_ridership_data'.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the input data flows listed above;
(b) when inputs of data are received, load their contents into the data store identified above;
(c) when the inputs requesting data are received, read the required data from the store and generate the output data flows identified above;
(d) periodically generate the data flows listed above requesting the current toll and parking lot prices plus transit fare data;
(e) be responsible for the management of the data in the store of the prices for services, using the most appropriate mechanism(s) such as RDBMS, for storing the data;
(f) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 1.0;
USR = 1.4;
USR = 1.4.0;
USR = 1.4.2;
USR = 1.4.2.3;
USR = 1.4.3;
USR = 1.4.3.5;
USR = 1.4.3.6;
USR = 3.0;
USR = 3.1;
USR = 3.1.2;
USR = 3.1.2.7;
USR = 3.1.4;
USR = 3.1.4.3;
USR = 3.1.5;
USR = 3.1.5.1;
USR = 3.1.5.1.1;
USR = 3.1.5.2;
USR = 3.1.5.3

Output Flow Dynamics Assumptions:

price_data_for_services = parking_lot_price_data+toll_price_data+transit_fare_data;
transit_fare_data_request = 12/(60*60*24*7*52)*PARKING_LOTS;
transit_fare_details = transit_fare_request;

parking_lot_charge_details = parking_lot_charge_request;
parking_lot_price_data_request = 12/(60*60*24*7*52)*PARKING_LOTS;
prices = request_prices;
toll_price_details = toll_price_request;
toll_price_data_request = 12/(60*60*24*7*52)*TOLL_PLAZAS;

7.4.3 Route Traveler Advanced Payments

Input Flows

advanced_traveler_charges_confirm
advanced_traveler_fares_confirm
advanced_traveler_tolls_confirm
traveler_advanced_payments_request

Output Flows

advanced_traveler_charges_request
advanced_traveler_fares_request
advanced_traveler_tolls_request
traveler_advanced_payments_confirm

Description:

Overview: This process shall be responsible for receiving a traveler's request for advanced payment (for tolls, parking lot charges, and/or transit fares) and routing it to the appropriate part of the Provide Electronic Payment Services function. The process shall also receive responses to the advanced payment requests and shall return them to the originating process.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the traveler advanced payments request flow listed above;
(b) when the flow in (a) is received, interrogate the data and generate the appropriate advanced fare, toll or charge request flow identified above;
(c) when all of the resulting responses have been received, generate the traveler advanced payments response flow listed above;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.2;
USR = 3.1.2.4;
USR = 3.1.2.6;
USR = 3.1.3;
USR = 3.1.3.2;
USR = 3.1.4;
USR = 3.1.4;
USR = 3.1.4.1;
USR = 3.1.4.2;

Output Flow Dynamics Assumptions:

advanced_traveler_charges_request = traveler_advanced_payments_request;
advanced_traveler_fares_request = traveler_advanced_payments_request;
advanced_traveler_tolls_request = traveler_advanced_payments_request;
traveler_advanced_payments_confirm = traveler_advanced_payments_request;

7.5.1 Provide Vehicle Payment Instrument Interface

Input Flows

cv_driver_enrollment_cost
driver_advanced_payment_for_map
fpi_driver_vehicle_input_credit_identity
fpi_transit_user_vehicle_input_credit_identity
transit_user_advanced_payment_on_vehicle

Output Flows

cv_driver_credit_identity
driver_credit_identity
tpi_debited_driver_payment_at_vehicle
tpi_debited_transit_user_payment_at_vehicle
transit_user_vehicle_credit_identity

Description:

Overview: This process shall be responsible for providing the interface through which driver credit identities and stored credit may be entered into the ITS from on-board vehicle tags. The types of vehicles from which data is collected shall include private cars or vans,
commercial vehicles, and transit vehicles, including those used for demand responsive transit services. This process shall also provide an interface through which the stored credit held by the tag can be debited for the payment of current or advanced tolls, plus advanced parking lot charges, and/or transit fares.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the appropriate one of the outputs identified above;
(c) when either of the input flows requesting a reduction in stored credit is received, generate the appropriate flow to reduce the credit currently stored on-board the payment instrument;
(d) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

cv_driver_credit_identity = fpi-driver_vehicle_input_credit_identity;
driver_credit_identity = fpi-driver_vehicle_input_credit_identity;
tpi-debited_driver_payment_at_vehicle = fpi-driver_vehicle_input_credit_identity;
tpi-debited_transit_user_payment_at_vehicle = fpi-transit_user_vehicle_input_credit_identity;
transit_user_vehicle_credit_identity = fpi-transit_user_vehicle_input_credit_identity;

7.5.2 Provide Transit User Roadside Payment Instrument Interface

Input Flows

fpi_transit_user_roadside_input_credit_identity
transit_user_advanced_payment_at_roadside

Output Flows

tpi_debited_transit_user_payment_at_roadside
transit_user_roadside_credit_identity

Description:

Overview: This process shall be responsible for providing the interface through which credit identities and stored credit values may be collected from tags being used by transit users. The process shall support the collection of this data at the roadside (which in this instance is a transit stop). Payments by the transit user for fares, other services, payment of advanced tolls, and/or parking lot charges shall be supported by the process. It shall also provide an interface through which stored credit held by the tag can be debited for the same types of payment.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the appropriate one of the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

transit_user_roadside_credit_identity = fpi-transit_user_roadside_input_credit_identity;
tpi-debited_transit_user_payment_at_roadside = fpi-transit_user_roadside_input_credit_identity;

7.5.3 Provide Personal Payment Instrument Interface

Input Flows

fpi_traveler_personal_input_credit_identity
traveler_personal_display_update_cost
traveler_personal_map_update_cost
traveler_personal_trip_costs

Output Flows

tpi_debited_payment_at_personal_device
traveler_personal_credit_identity

Description:

Overview: This process shall be responsible for providing the interface through which credit identity or stored credit may be collected from the tag being used by a traveler with a personal device. The process shall support the collection of this data from any location in which the device (and hence the tag) is being used. It shall provide an interface through which the credit identity can be used for the payment of advanced tolls, parking lot charges, transit fares, display updates, and/or map updates. The process shall also enable the stored credit value on the tag to be used for the same purposes.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the appropriate one of the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

tpi-debited_payment_at_personal_device = fpi-traveler_personal_input_credit_identity;
traveler_personal_credit_identity = fpi-traveler_personal_input_credit_identity;

7.5.4 Provide Commercial Fleet Payment Instrument Interface

Input Flows

cf_manager_enrollment_cost
fpi_commercial_manager_input_credit_identity

Output Flows

cf_manager_credit_identity
tpi_debited_commercial_manager_payment

Description:

Overview: This process shall be responsible for providing the interface through which credit identity or stored credit values may be collected from the tags used by commercial fleet managers. The process shall support the use of the tag data to complete payment for the filing of electronic credentials and tax information that will enable a commercial vehicle to be cleared to travel within the geographic area served by a particular jurisdictional authority. This process shall also enable the stored credit value to be debited as an alternative method of payment.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the appropriate one of the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

cf_manager_credit_identity = fpi-commercial_manager_input_credit_identity;
tpi-debited_commercial_manager_payment = fpi-commercial_manager_input_credit_identity;

7.5.5 Provide Traveler Kiosk Payment Instrument Interface

Input Flows

fpi_traveler_roadside_input_credit_identity
traveler_roadside_trip_costs

Output Flows

tpi_debited_traveler_payment_at_roadside
traveler_roadside_credit_identity

Description:

Overview: This process shall be responsible for providing the interface through which credit identities and stored credit values may be collected from payment instruments being used by travelers. The process shall support the collection of data at the roadside (which in this instance is a kiosk) and use this data for payments needed to confirm a traveler's trip. Payments supported by the process shall include those for advanced tolls, parking lot charges, transit fares, and/or other (yellow pages) services. It shall also provide an interface through which the stored credit held by the tag can be debited for the same types of payment.

Data Flows: All input data flows are unsolicited and all output flows are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received, generate the appropriate one of the outputs identified above;
(c) all input and output flows must be encrypted in such a way that it is not possible to determine the payment information being transmitted, using any form of digital or analog techniques.

User Service Requirements:

USR = 3.0;
USR = 3.1;
USR = 3.1.0;
USR = 3.1.3;
USR = 3.1.3.1;

Output Flow Dynamics Assumptions:

traveler_roadside_credit_identity = fpi-traveler_roadside_input_credit_identity;
tpi-debited_traveler_payment_at_roadside = fpi-traveler_roadside_input_credit_identity;

8.1 Get Archive Data

Input Flows

collected_roadside_data
cv_archive_data
em_archive_data
emissions_archive_data
fcm_c_and_m_archive_data
fifd_intermodal_archive_data
fmtsp_multimodal_archive_data
fmup_map_archive_data
fods_other_data_source_archive_data
fws_weather_archive_data
import_administration_request
parking_archive_data
toll_archive_data
traffic_management_archive_data
transit_archive_data
traveler_archive_data

Output Flows

collected_roadside_data_status
cv_archive_request
cv_archive_status
em_archive_request
em_archive_status
emissions_archive_request
emissions_archive_status
import_administration_status
parking_archive_request
parking_archive_status
retrieved_archive_data
tcm_c_and_m_archive_request
tcm_c_and_m_archive_status
tifd_intermodal_archive_request
tifd_intermodal_archive_status
tmtsp_multimodal_archive_request
tmtsp_multimodal_archive_status
tmup_map_archive_request
tmup_map_archive_status
tods_other_data_source_archive_request
tods_other_data_source_archive_status
toll_archive_request
toll_archive_status
traffic_management_archive_request
traffic_management_archive_status
transit_archive_request
transit_archive_status
traveler_archive_request
traveler_archive_status
tws_weather_archive_request
tws_weather_archive_status

Description:

Overview: This process shall collect data from each major function within ITS and external sources for archive purposes that may not exist within current ITS data sources. This process shall respond to requests from the Manage Archive Data Administrator Interface process to import data or data catalogs. This process shall send requests for data or a catalog of available data to the other functions and terminators, either a subscription for data or a one-time request. This process shall receive meta data along with the data to describe the conditions under which the data was collected or any other information about the operational data. When data is received this process shall perform quality checks such as range validation or reformat the data as necessary to meet the archive schema. This process shall execute methods on the incoming data such as cleansing, summarizations, aggregations, or transformations applied to the data before it is stored in the archive. Any changes made to the data shall be recorded in the meta-data stored in the archive to assist in the reconstruction of the original data if possible. This process shall receive inputs from the Manage Archive Data Administrator Interface that contain the parameters for managing the processing on the data. This process forwards the collected onto the Manage Archive function along with updated meta-data and a record of any methods applied to the incoming data. This process shall also support the notification of the operational source functions of any errors that may be present in the data that could be caused by equipment failures or a transmission error.

Data Flows: The input data flow import_administration_request is unsolicited. The other input data flows of the form xxx_archive_data are solicited by the process during a one-time request operation; subsequent data sent as part of a subscription operation shall be received as unsolicited input. All outputs from this process are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the unsolicited data input import_administration_request is received, the process shall generate the solicited output flow of the form xxx_archive_request where xxx is the source for the data requested;
(c) when the data xxx_archive_data is received, either unsolicited as part of a subscription arrangement or solicited in response to the request issued in (b), the process shall received the data and format it per the information contained in the import_administration_request flow received in (b);
(d) when the input data has been formatted the process shall send the data to the Manage Archive function;
(e) the process shall update the meta data of the received data to describe any formatting steps performed in (c);
(f) the process shall generate the solicited output import_administration_status to inform the Manage Archive Administrator Interface function of the status of the import process and to include the catalog of data requested and available from the source function;
(g) when data is received from an input source, the process shall generate solicited output xxx_archive_status to notify the source of any errors in the received data.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.1;
USR = 7.1.1.1;
USR = 7.1.1.3;
USR = 7.1.2;
USR = 7.1.2.1;
USR = 7.1.2.1.1;
USR = 7.1.2.1.2;
USR = 7.1.2.1.3;
USR = 7.1.2.1.3(a);
USR = 7.1.2.1.3(b);
USR = 7.1.2.1.3(c);
USR = 7.1.3.1.1(e);
USR = 7.1.3.1.10;
USR = 7.1.3.1.11;
USR = 7.1.3.1.2;
USR = 7.1.3.1.3;
USR = 7.1.3.1.3(a);
USR = 7.1.3.1.3(b);
USR = 7.1.3.1.3(c);
USR = 7.1.3.1.3(d);
USR = 7.1.3.1.3(e);
USR = 7.1.3.1.4;
USR = 7.1.3.1.4(a);
USR = 7.1.3.1.4(b);
USR = 7.1.3.1.4(c);
USR = 7.1.3.1.4(d);
USR = 7.1.3.1.4(e);
USR = 7.1.3.1.4(f);
USR = 7.1.3.1.4(g);
USR = 7.1.3.1.5;
USR = 7.1.3.1.5(a);
USR = 7.1.3.1.5(b);
USR = 7.1.3.1.5(c);
USR = 7.1.3.1.5(d);
USR = 7.1.3.1.5(e);
USR = 7.1.3.1.5(f);
USR = 7.1.3.1.5(g);
USR = 7.1.3.1.5(h);
USR = 7.1.3.1.6;
USR = 7.1.3.1.6(a);
USR = 7.1.3.1.6(b);

USR = 7.1.3.1.6(c);
USR = 7.1.3.1.6(d);
USR = 7.1.3.1.6(e);
USR = 7.1.3.1.6(f);
USR = 7.1.3.1.7;
USR = 7.1.3.1.7(a);
USR = 7.1.3.1.7(b);
USR = 7.1.3.1.8;
USR = 7.1.3.1.8(a);
USR = 7.1.3.1.8(b);
USR = 7.1.3.1.8(c);
USR = 7.1.3.1.8(d);
USR = 7.1.3.1.8(e);
USR = 7.1.3.1.8(f);
USR = 7.1.3.1.8(g);
USR = 7.1.3.1.8(h);
USR = 7.1.3.1.9;
USR = 7.1.3.1.9(a);
USR = 7.1.3.1.9(b);
USR = 7.1.3.1.9(c);
USR = 7.1.3.1.9(d);
USR = 7.1.3.1.9(e);
USR = 7.1.3.1.9(f);
USR = 7.1.3.2;
USR = 7.1.3.3;
USR = 7.1.3.4;
USR = 7.1.3.5;
USR = 7.1.3.5.1;
USR = 7.1.3.5.2;
USR = 7.1.3.6;
USR = 7.1.3.7;
USR = 7.1.3.8;
USR = 7.1.3.9;
USR = 7.1.4;
USR = 7.1.4.1;
USR = 7.1.4.1.1;
USR = 7.1.4.1.2;
USR = 7.1.4.1.3;
USR = 7.1.4.2;
USR = 7.1.4.2(c);
USR = 7.1.4.4;
USR = 7.1.4.4(a);
USR = 7.1.4.4(b);
USR = 7.1.4.4(c);
USR = 7.1.4.5;
USR = 7.1.6;
USR = 7.1.6.2;
USR = 7.1.6.2.1;
USR = 7.1.2.1.4;
USR = 7.1.2.1.5;
USR = 7.1.2.1.5(a);
USR = 7.1.2.1.5(b);
USR = 7.1.2.2;
USR = 7.1.2.3;
USR = 7.1.2.4;
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.1;
USR = 7.1.3.1.1(a);
USR = 7.1.3.1.1(b);
USR = 7.1.3.1.1(c);
USR = 7.1.3.1.1(d);

Output Flow Dynamics Assumptions:

import_administration_status = 2/(60*60*24);
retrieved_archive_data = 1; collected_roadside_data_status = 1/60;

tods-other_data_source_archive_status = 1/60;
tods-other_data_source_archive_request = 1/60;
tmup-map_archive_status = 1/60;
tmup-map_archive_request = 1/60;
tcm-c_and_m_archive_request = 1/60;
tcm-c_and_m_archive_status = 1/60;
tifd-intermodal_archive_status = 1/60;
tifd-intermodal_archive_request = 1/60;
tws-weather_archive_status = 1/60;
tws-weather_archive_request = 1/60;
tmtsp-multimodal_archive_status = 1/60;
tmtsp-multimodal_archive_request = 1/60;
toll_archive_status = 1/60;
toll_archive_request = 1/60;
traveler_archive_status = 1/60;
traveler_archive_request = 1/60;
em_archive_status = 1/60;
em_archive_request = 1/60;
transit_archive_status = 1/60;
transit_archive_request = 1/60;
cv_archive_status = 1/60;
cv_archive_request = 1/60;
emissions_archive_status = 1/60;
emissions_archive_request = 1/60;
parking_archive_status = 1/60;
parking_archive_request = 1/60;
traffic_management_archive_status = 1/60;
traffic_management_archive_request = 1/60;

8.2 Manage Archive

Input Flows

analyze_archive_data_request
archive_administration_request
archive_data
archive_data_product_request
global_schema
government_report_data_request
other_archive_data_input
other_archive_data_request_input
retrieved_archive_data

Output Flows

archive_administration_data
archive_data
archive_data_for_analysis
archive_data_product
government_report_data
local_schema
other_archive_data
other_archive_data_request

Description:

Overview: This process shall store the collected and formatted data in a permanent archive data store. This process shall receive the formatted data from the Get Archive Data function accompanied by any updates to the meta data that would describe the formatting operations performed on the data as it was imported. This process shall respond to requests from the administrator interface function to maintain the schema of the archive data, set update frequencies, backup schedules, user authentication schemes, cleansing algorithms. This process shall provide the administrator interface function with status of the data quality in the archive, frequency reports on use of the archive, updates to the measure of the volume of the data and other data archive metrics. This process shall receive inputs from the Coordinate Archives function to provide data and information about the archive schema to other archives. In turn the process shall receive data and schema of other archives to use to build a global schema. The process shall use the global schema to support requests from user systems for data that may be spread across multiple archives. The process shall maintain the access privileges information for the data held in the archive to maintain the security of the archive. The process shall employ such techniques as necessary to maintain the integrity of the data and ensure no data is lost from the archive. The process shall respond to requests for data to support user data products, user analysis, and inputs to government reporting systems. The process shall respond to such request by authenticating the originator of the request and providing the data that is available. The process shall also be capable of providing a sample or catalog of data contained within the archive to support the user requests.

Data Flows: All inputs to this process are unsolicited and all outputs from this process are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the unsolicited data input retrieved_archive_data is received, the process shall update the data store;
(c) when the unsolicited data input for administration_request is received, the process shall respond with the solicited data output administration_status;
(d) when the input flows requesting data from the archive are received, the process shall authenticate the user can access to the data, determine the location of the data, whether local or in another archive, and generate the requested data output.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.1;
USR = 7.1.1.1;
USR = 7.1.1.2;
USR = 7.1.1.3;
USR = 7.1.1.4;
USR = 7.1.1.4.1;
USR = 7.1.1.4.2;
USR = 7.1.1.4.3;
USR = 7.1.1.4.4;
USR = 7.1.2.3;
USR = 7.1.4;
USR = 7.1.4.1;
USR = 7.1.4.1.1;

USR = 7.1.4.1.2;
USR = 7.1.4.1.3;
USR = 7.1.4.2;
USR = 7.1.4.2(a);
USR = 7.1.4.2(b);
USR = 7.1.4.2(c);
USR = 7.1.4.2(d);
USR = 7.1.4.3;
USR = 7.1.4.4;
USR = 7.1.4.5;
USR = 7.1.5;
USR = 7.1.5.1;
USR = 7.1.5.1(a);
USR = 7.1.5.1(b);
USR = 7.1.5.1(c);
USR = 7.1.5.1(d);
USR = 7.1.5.2;
USR = 7.1.5.2.3;
USR = 7.1.5.2.4;
USR = 7.1.6;
USR = 7.1.6.2;
USR = 7.1.6.2.1;
USR = 7.1.6.2.2;
USR = 7.1.6.3;
USR = 7.1.6.3.1;
USR = 7.1.6.3.2;
USR = 7.1.6.3.3;

Output Flow Dynamics Assumptions:

archive_administration_request = 2/(60*60*24);
archive_administration_data = 2/(60*60*24);
local_schema = 1/60;
other_archive_data = 1/60;
other_archive_data_request = 1/60;
government_report_data = 1/(60*60);
archive_data_product = 1;
archive_data_for_analysis = 1/60;

8.3 Manage Archive Data Administrator Interface

Input Flows

archive_administration_data
collection_administration_status
fada_archive_administration_requests
import_administration_status
on_demand_archive_request

Output Flows

archive_administration_request
archive_request_confirmation
collection_administration_request
import_administration_request
tada_archive_administration_data

Description:

Overview: This process shall interface with the Archive Data Administrator terminator and receive inputs from the administrator concerning the management and administration of the archive. The process shall establish user authentication controls for the archive and send the information to the Manage Archive function. The process shall maintain the schema of the archive, including the data and meta data contained within the archive data. Updates to the schema shall be distributed to the Manage Archive function as well as the Get Archive Data function. The process shall send the parameters and requests to the Get Archive Data function to control what data is imported into the archive and how the data is to be formatted when it is received. The parameters sent shall include such things as the schema, data format, methods to apply to the data, cleansing parameters, quality metrics, and checking specifications. The process shall send requests to the Get Archive Data function for new data or a catalog of data that may be available. The process shall respond to requests from the Manage On Demand Archive Requests function by making requests of the Get Archive Data function to establish the source and identity of the data that may exist in ITS or non-ITS sources. Then the process shall respond to the user request with the confirmation that the request can be satisfied and specifications about the data once it is imported. In cases where the Manage Archive function will be managing a roadside data collection function, this process shall initiate and control the function by sending commands and requests to the Manage Roadside Data Collection function. This process receives the status from the other functions within Manage Archived Data and presents them to the administrator.

Data Flows: All input flows are solicited with the exception of 'fada-archive_administration_requests' and 'on_demand_archive_requests' which are unsolicited. All outputs are solicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received from the administrator, generate the appropriate output data flow;
(c) when the input is received from the on demand request function, generate the output to the Get Archive Data function;
(d) when status flows are received, generate the output for the administrator;
(e) all input and output flows regarding the security of the archive must be encrypted in such a way that it is not possible to determine the user identity or password authentication methods for any user.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.1.4;
USR = 7.1.1.4.1;
USR = 7.1.1.4.2;
USR = 7.1.1.4.3;
USR = 7.1.1.4.4;
USR = 7.1.2;
USR = 7.1.2.1;
USR = 7.1.2.1.1;
USR = 7.1.2.1.2;
USR = 7.1.2.1.4;
USR = 7.1.2.1.5;
USR = 7.1.2.1.5(a);
USR = 7.1.2.1.5(b);
USR = 7.1.2.2;
USR = 7.1.2.4;
USR = 7.1.3;
USR = 7.1.3.1;

USR = 7.1.3.2;
USR = 7.1.3.3;
USR = 7.1.3.4;
USR = 7.1.3.5;
USR = 7.1.3.5.1;
USR = 7.1.3.5.2;
USR = 7.1.3.6;
USR = 7.1.3.7;
USR = 7.1.3.8;
USR = 7.1.4.1;
USR = 7.1.4.1.1;
USR = 7.1.4.1.2;
USR = 7.1.4.1.3;
USR = 7.1.4.2;
USR = 7.1.4.2(a);
USR = 7.1.4.2(b);
USR = 7.1.4.2(c);
USR = 7.1.4.2(d);
USR = 7.1.4.4;
USR = 7.1.4.4(a);
USR = 7.1.4.4(b);
USR = 7.1.4.4(c);
USR = 7.1.4.5;
USR = 7.1.5.3;
USR = 7.1.6;
USR = 7.1.6.1;
USR = 7.1.6.1.1;
USR = 7.1.6.2;
USR = 7.1.6.2.1;
USR = 7.1.6.2.2;
USR = 7.1.6.3;
USR = 7.1.6.3.1;
USR = 7.1.6.4;

Output Flow Dynamics Assumptions:

archive_request_confirmation = fada-archive_administration_requests;
archive_administration_request = fada-archive_administration_requests;
import_administration_request = fada-archive_administration_requests;
collection_administration_request = fada-archive_administration_requests;
tada-archive_administration_data = fada-archive_administration_requests;

8.4 Coordinate Archives

Input Flows

foa_archive_coordination_data
local_schema
other_archive_data
other_archive_data_request

Output Flows

global_schema
other_archive_data_input
other_archive_data_request_input
toa_archive_coordination_data

Description:

Overview: This process shall coordinate the information exchange between different Manage Archived Data functions represented through the Other Archives terminator. This process shall allow other archives to share data collected by other archive functions to share the data in response to local requests from users systems. This process shall use data collected from different archives to build a set of global schema which the data archive definitions for the local archive plus any archives known to the local archive. This process shall provide the global schema to the local Manage Archive function. This process shall receive the schema of the local archive to share with other archive functions. This process shall provide data to those other archives when requested. This process shall support analysis, data fusion, and data mining of archived information across geographically dispersed archives.

Data Flows: All input and output flows are solicited with the exception of other_archive_data_request and foa-archive_coordination_data which can be unsolicited.

Functional Requirements: This process shall meet the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when the input in (a) is received, send the data flow to the other archive to request data be provided in response to a local user systems request or to the Manage Archive function in response to the other archives request;
(c) when are response to the request in (b) is received, forward the data from other archives to the Manage Archive function and forward the local data to the other archives;
(d) when local schema arrives from the Manage Archive function update the other archives with the definition of the local archive schema;
(e) when schema about other archives is received update the Manage Archive function with the global_schema flow.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.4.4;
USR = 7.1.5;
USR = 7.1.5.1;
USR = 7.1.5.1(a);
USR = 7.1.5.1(b);
USR = 7.1.5.1(c);
USR = 7.1.5.1(d);
USR = 7.1.5.2;
USR = 7.1.5.2.3;
USR = 7.1.5.2.4;
USR = 7.1.6;
USR = 7.1.6.2;
USR = 7.1.6.2.2;
USR = 7.1.6.3;
USR = 7.1.6.3.1;

Output Flow Dynamics Assumptions:

toa-archive_coordination_data = 1/(60*60*24);
global_schema = 1/(60*60*24);
other_archive_data_request_input = 1/(60*60*24);
other_archive_data_input = 1/(60*60*24);

8.5 Process Archived Data User System Requests

Input Flows

archive_data_product
fadu_archive_data_product_request
ffi_archive_payment_confirm

Output Flows

archive_data_product_request
tadu_archive_data_product
tfi_archive_payment_request

Description:

Overview: This process shall monitor the archive data user systems interface for requests for data from the archive. This process shall support requests from users involved in planning, research, safety, as well as operations of transportation functions. This process shall receive requests for data and catalogs of data that may be contained in the archive. This process shall translate the requests into a format that can be understood by the Manage Archive function to retrieve data from the archive. When data or a catalog of data is received from the archive, this process shall generate the requested data product for the users systems. For archive data requiring financial payment this archive process the financial requests and manages an interface to a Finanicial Institution.

Data Flows: All input and output flows are solicited with the exception of fadu-archive_data_product_request which is unsolicited.

Functional Requirements: This process shall satisfy the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when a request is received from a user system, generate the request outpu to forward
the request to the Manage Archive function;
(c) when the data is received from the archive, either the catalog of data, the data itself,
or meta data; immediately generate the output to the user system;
(d) before output, the process shall put the data into a format that is easily read and interpreted by external processes.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.1.4.1;
USR = 7.1.1.4.3;
USR = 7.1.1.4.4;
USR = 7.1.2;
USR = 7.1.2.5;
USR = 7.1.3.7;
USR = 7.1.4.4;
USR = 7.1.4.4(c);
USR = 7.1.5;
USR = 7.1.5.1;
USR = 7.1.5.1(a);
USR = 7.1.5.1(b);
USR = 7.1.5.1(c);
USR = 7.1.5.1(d);
USR = 7.1.5.2;
USR = 7.1.5.2.2;
USR = 7.1.5.2.3;
USR = 7.1.5.2.4;
USR = 7.1.6;
USR = 7.1.6.1;
USR = 7.1.6.1.1;
USR = 7.1.6.2;
USR = 7.1.6.2.2;
USR = 7.1.6.3;
USR = 7.1.6.3.1;
USR = 7.1.6.3.3;

Output Flow Dynamics Assumptions:

tadu-archive_data_product = 1/60
archive_data_product_request = 1/60
tfi-archive_payment_request = 1/(60*60*24);

8.6 Analyze Archive

Input Flows

archive_data_for_analysis
fadu_archive_analysis_request
ffi_archive_analysis_payment_confirm

Output Flows

analyze_archive_data_request
tadu_archive_analysis_results
tfi_archive_analysis_payment_request

Description:

Overview: This process shall support the interface with Archive Data User Systems for requests for analysis of the archive data. This process shall support analysis products that can provide users with the ability to perform activities such as data mining, data fusion, summarizations, aggregations, and recreation from archive data. This process shall receive the users systems requests and develop the request that the Manage Archive function can process to retrieve the data from the archive. This process shall be able to respond to users systems requests for a catalog of the analysis products available. When data and meta data are returned from the archive and the analysis is performed this process shall produce the output for the Archive Data User Systems terminator. For archive data requiring financial payment this archive process the financial requests and manages an interface to a Finanicial Institution.

Data Flows: All input and output flows are solicited with the exception of fadu-archive_analysis_request which is unsolicited.

Functional Requirements: This process shall satisfy the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when a request is received from a user system, generate the request output to forward the request to the Manage Archive function;
(c) when the data is received from the archive, either the catalog of data, the data itself, or meta data; immediately perform the analysis requested and generate the output to the user system;
(d) before output, the process shall put the data into a format that is easily read and interpreted by external processes.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.1.4.1;
USR = 7.1.1.4.3;
USR = 7.1.1.4.4;
USR = 7.1.1.5;
USR = 7.1.2;
USR = 7.1.2.6;
USR = 7.1.3.7;
USR = 7.1.4.4;
USR = 7.1.5;
USR = 7.1.5.1;
USR = 7.1.5.1(a);
USR = 7.1.5.1(b);
USR = 7.1.5.1(c);
USR = 7.1.5.1(d);
USR = 7.1.5.2;
USR = 7.1.5.2.1;
USR = 7.1.5.2.1(a);
USR = 7.1.5.2.1(b);
USR = 7.1.5.2.1(c);
USR = 7.1.5.2.1(d);
USR = 7.1.5.2.3;
USR = 7.1.5.2.4;
USR = 7.1.6;
USR = 7.1.6.1;
USR = 7.1.6.1.1;
USR = 7.1.6.2;
USR = 7.1.6.2.2;

USR = 7.1.6.3;
USR = 7.1.6.3.1;

<u>Output Flow Dynamics Assumptions:</u>

analyze_archive_data_request = 1/(60*60);
tadu-archive_analysis_results = 1/(60*60);
tfi-archive_analysis_payment_request = 1/(60*60*24

8.7 Process On Demand Archive Requests

Input Flows
archive_request_confirmation
fadu_on_demand_archive_request

Output Flows
on_demand_archive_request
tadu_on_demand_confirmation

Description:
Overview: This process shall receive requests for data to be imported into the archive that is not already in the archive. The process shall forward the request to the Manage Archive Data Administrator Interface function for the administrator to handle the user request. The process shall receive the response from the administrator and forward the information to the Archive Data User System.

Data Flows: All input and output flows are solicited with the exception of fadu-on_demand_archive_request which is unsolicited.

Functional Requirements: This process shall satisfy the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input flows listed above;
(b) when a request is received from a user system, generate the request output to forward the request to the Manage Archive Data Administrator Interface function;
(c) when the response is received from the administrator, generate the output to the user system;
(d) before output, the process shall put the data into a format that is easily read and interpreted by external processes.

User Service Requirements:
USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.2;
USR = 7.1.2.2;
USR = 7.1.4.4;
USR = 7.1.4.4(a);
USR = 7.1.4.4(b);
USR = 7.1.4.4(c);
USR = 7.1.6;
USR = 7.1.6.1;
USR = 7.1.6.1.1;
USR = 7.1.6.2;
USR = 7.1.6.2.2;
USR = 7.1.6.3;
USR = 7.1.6.3.1;
USR = 7.1.6.4;
USR = 7.1.6.4.1;
USR = 7.1.6.4.1(a);
USR = 7.1.6.4.1(b);
USR = 7.1.6.4.1(c);
USR = 7.1.6.4.1(d);
USR = 7.1.6.4.1(e);
USR = 7.1.6.4.1(f);
USR = 7.1.6.4.1(g);
USR = 7.1.6.4.2;
USR = 7.1.6.4.2(a);
USR = 7.1.6.4.2(b);
USR = 7.1.6.4.2(c);
USR = 7.1.6.4.2(d);
USR = 7.1.6.4.3;
USR = 7.1.6.4.3(a);
USR = 7.1.6.4.3(b);
USR = 7.1.6.4.3(c);
USR = 7.1.6.4.4;

Output Flow Dynamics Assumptions:
tadu-on_demand_confirmation = 1/(60*60*24);
on_demand_archive_request = 1/(60*60*24);

8.8 Prepare Government Reporting Inputs

Input Flows

fgrs_government_data_report_request
government_report_data

Output Flows

government_report_data_request
tgrs_government_data_report_input

Description:

Overview: This process shall support the preparation of inputs to reporting systems of the federal or state governments that require data from the ITS archive. This process shall respond to requests from the Government Reporting Systems terminator for data from the archive and generate the request in a form understood by the Manage Archive function. The data and any meta data necessary shall be returned from the Manage Archive function. This process shall receive the data and format it as requested and send it to the Government Reporting Systems terminator where it may be combined with other data before final submission.

Data Flows: All input and output flows are solicited with the exception of fgrs-government_data_report_request which is unsolicited.

Functional Requirements: This process shall satisfy the following functional requirements:
(a) continuously monitor for receipt of the unsolicited input listed above;
(b) upon receipt of the input listed in (a), generate the data request to the Manage Archive function to provide the data required from the archive;
(c) upon receipt of the returned data requested in (b), generate the output to the Government Reporting Systems terminator.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.5;
USR = 7.1.5.2;
USR = 7.1.5.2.5;
USR = 7.1.5.2.5(a);
USR = 7.1.5.2.5(b);
USR = 7.1.5.2.5(c);
USR = 7.1.5.2.5(d);
USR = 7.1.5.2.5(e);
USR = 7.1.5.2.5(f);
USR = 7.1.5.2.5(g);
USR = 7.1.5.2.5(h);
USR = 7.1.5.2.5(i);
USR = 7.1.5.2.5(j);
USR = 7.1.6;
USR = 7.1.6.2;
USR = 7.1.6.2.2;
USR = 7.1.6.3;
USR = 7.1.6.3.1;

Output Flow Dynamics Assumptions:

government_report_data_request = 1/(60*60);
tgrs-government_data_report_input = 1/(60*60);

8.9 Manage Roadside Data Collection

Input Flows

collected_roadside_data_status
collection_administration_request
roadside_archive_data

Output Flows

collected_roadside_data
collection_administration_status
roadside_archive_control

Description:

Overview: This process shall manage the collection of archive data directly from collection equipment located at the roadside. This process shall collect traffic information as well as environmental or other information that may be collected by roadside devices. This process shall respond to requests from the Manage Archive Data Administer Interface process to input the parameters that control the collection process. The request for data and control parameters shall be sent to the Manage Traffic function where the information is collected and returned. This process shall forward the data onto the Get Archive Data function for import into the archive. The Get Archive Data function shall be able to return status about the imported data. This process shall use the status information to adjust the collection function and report back to the administrator function.

Data Flows: All input flows are unsolicited with the exception of collected_roadside_data_status and roadside_archive_data which are unsolicited. All outputs are solicited.

(a) continuously monitor for receipt of the unsolicited input flow listed above;
(b) when the input is received from the administrator, generate the appropriate output data flow;
(c) when data is received from the roadside_archive_data, check the data for errors and forward the data to the Get Archive Function on output collected_roadside_data;
(d) update the collection_administration_status upon receipt of the archive data and the status from the Get Archive Data function.

User Service Requirements:

USR = 7.0;
USR = 7.1;
USR = 7.1.0;
USR = 7.1.2;
USR = 7.1.2.1;
USR = 7.1.2.1.1;
USR = 7.1.2.1.2;
USR = 7.1.2.1.3;
USR = 7.1.2.1.3(a);
USR = 7.1.2.1.3(b);
USR = 7.1.2.1.3(c);
USR = 7.1.3;
USR = 7.1.3.1;
USR = 7.1.3.1.1;
USR = 7.1.3.1.1(a);
USR = 7.1.3.1.1(c);
USR = 7.1.3.1.3;
USR = 7.1.3.1.3(e);
USR = 7.1.3.1.7;
USR = 7.1.3.1.7(a);

Output Flow Dynamics Assumptions:

roadside_archive_control = collection_administration_requests;
collected_roadside_data = roadside_archive_data;
collection_administration_status = collection_administration_requests;

9 Satisfy Implementation Requirements

Output Flows

Description:

Overview: This process represents the physical implementation of functions and communications links that are required by the architecture. It has no data flows or logical functions but is needed to meet the User Service Requirements (USR's).

Data Flows:

Functional Requirements: This process shall meet the following functional requirements:

User Service Requirements:

USR = 1.4.2.2;
USR = 2.1.4.5;
USR = 2.3.4.1;
USR = 4.5.3.5;

Output Flow Dynamics Assumptions:

output_data_flow = 0;